TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI UNTUK KLASIFIKASI KOMENTAR NETIZEN PADA MEDIA SOSIAL PEMERINTAH DAERAH DI INDONESIA MENGGUNAKAN ALGORITMA RANDOM FOREST

DEVELOPING AN APPLICATION FOR NETIZEN'S COMMENTS CLASSIFICATION ON SOCIAL MEDIA OF INDONESIA LOCAL GOVERNMENTS USING RANDOM FOREST ALGORITHM

MUHAMMAD FIKRY HAZMI
NRP 052 1144 0000 143

Dosen Pembimbing:
Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

TUGAS AKHIR – KS141501

RANCANG BANGUN APLIKASI UNTUK KLASIFIKASI KOMENTAR NETIZEN PADA MEDIA SOSIAL PEMERINTAH DAERAH DI INDONESIA MENGGUNAKAN ALGORITMA RANDOM FOREST

MUHAMMAD FIKRY HAZMI
NRP 052 1144 0000 143

Dosen Pembimbing:
Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

Halaman ini sengaja dikosongkan

FINAL PROJECT – KS141501

DEVELOPING AN APPLICATION FOR NETIZEN'S COMMENTS CLASSIFICATION ON SOCIAL MEDIA OF INDONESIA LOCAL GOVERNMENTS USING RANDOM FOREST ALGORITHM

MUHAMMAD FIKRY HAZMI
NRP 052 1144 0000 143

Supervisor:
Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D

INFORMATION SYSTEMS DEPARTMENT
Faculty of Information and Communication Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

RANCANG BANGUN APLIKASI UNTUK KLASIFIKASI KOMENTAR NETIZEN PADA MEDIA SOSIAL PEMERINTAH DAERAH DI INDONESIA MENGGUNAKAN ALGORITMA RANDOM FOREST

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD FIKRY HAZMI
NRP. 052 1144 0000 143

Surabaya, 16 Juli 2018

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP 19650310 199102 1 001

Halaman ini sengaja dikosongkan

LEMBAR PERSETUJUAN

RANCANG BANGUN APLIKASI UNTUK KLASIFIKASI KOMENTAR NETIZEN PADA MEDIA SOSIAL PEMERINTAH DAERAH DI INDONESIA MENGGUNAKAN ALGORITMA RANDOM FOREST

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:

MUHAMMAD FIKRY HAZMI
NRP. 052 1144 0000 143

Disetujui Tim Penguji: Tanggal Ujian: 6 Juli 2018
Periode Wisuda: September 2018

Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D **(Pembimbing I)**

Faizal Johan Atletiko, S.Kom, M.T **(Penguji I)**

Renny Pradina Kusumawardani, S.T, M.T, SCJP **(Penguji II)**

Halaman ini sengaja dikosongkan

RANCANG BANGUN APLIKASI UNTUK KLASIFIKASI KOMENTAR NETIZEN PADA MEDIA SOSIAL PEMERINTAH DAERAH DI INDONESIA MENGGUNAKAN ALGORITMA RANDOM FOREST

Nama Mahasiswa	**:**	**Muhammad Fikry Hazmi**
NRP	**:**	**052 1144 0000 143**
Departemen	**:**	**Sistem Informasi FTIK-ITS**
Pembimbing	**:**	**Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D**

ABSTRAK

Perkembangan teknologi informasi dan komunikasi (TIK) mendorong pemerintah untuk menerapkan e-government. E-Government diyakini dapat memberikan dampak yang luas apabila dijalankan dengan baik. Media sosial dipilih sebagai jembatan komunikasi antara masyarakat dan pemerintah dengan pengguna aktif di Indonesia hampir 50% dari total populasi berdasarkan statistik dari Digital Global Statistic. Salah satu upaya yang dapat dilakukan untuk meningkatkan partisipasi publik adalah dengan membangun komunikasi dua arah yang efektif dengan masyarakat melalui media sosial. Komentar dalam media sosial merupakan salah satu bentuk keterlibatan masyarakat terhadap pemerintahan. Berdasarkan kasus tersebut, dibutuhkan sebuah platform yang mampu mengklasifikasikan dan memberikan informasi visual terhadap topik pembicaraan masyarakat pada kiriman di media sosial pemerintahan secara real-time dan Random forest dipilih sebagai metode klasifikasi. Adapun data yang digunakan diambil dari akun Facebook, Twitter, dan Youtube milik pemerintah daerah. Setelah data dididapatkan serta dianalisa menggunakan Random Forest kemudian dilakukan visualisasi terhadap kategori komentar. Platform ini menggunakan Kafka sebagai data pipeline dan menggunakan Spark untuk membantu proses machine learning. Melalui proses pembuatan model,

telah didapatkan hasil akurasi terbaik adalah 74,25% dengan menggunakan parameter num of trees dan max depth sebesar 100 dan 30. Selain itu pada proses aplikasi streaming, rata-rata Processing Time adalah 20,385 detik dan rata-rata Total Delay adalah 20,854 detik untuk setiap batch.

Kata Kunci: E-Government, Klasifikasi, Random Forest, Real-Time

DEVELOPING AN APPLICATION FOR NETIZEN'S COMMENTS CLASSIFICATION ON SOCIAL MEDIA OF INDONESIA LOCAL GOVERNMENTS USING RANDOM FOREST ALGORITHM

Name	**:**	**Muhammad Fikry Hazmi**
NRP	**:**	**052 1144 0000 143**
Department	**:**	**Sistem Informasi FTIK-ITS**
Supervisor	**:**	**Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D**

ABSTRACT

The development of information technology and communication has encouraged the government to apply e-government. They believe it could give a broad impact when it well implemented. Social media was chosen to be the communication tool between the society and the government. Based on statistics from the Digital Global Statistic, its active users in Indonesia reach almost 50% of the total population. One of the efforts that could be done to increase public participation is to build an effective two-way communication with society through the social media. Comments in the social media are one form of social participation to the government. Based on that case, we need a platform that could classify and give visual information about topics those spread widely in the society, from the posts they put on the government's social media accounts, in real time. To satisfy this, we chose Random Forest to be the method that used in the classification process. The data we used in this research is taken from Facebook, Twitter, and Youtube account of each local government. After gathering the data and processed it with random forest, then we visualized it in the comment's categories. This platform uses Kafka as a pipeline data and uses Spark to help machine learning process. Through the model development, we got 74,25% as the best accuracy with parameters; the num of trees and max depth are 100 and 30

respectively. Moreover, at the streaming application process, the average Processing Time is 20.385 seconds and the average Total Delay is 20.854 seconds for each batch.

Keywords: E-Government, Classification, Random Forest, Real-Time

KATA PENGANTAR

Puji dan syukur penulis tuturkan ke hadirat Allah SWT, Tuhan Semesta Alam yang telah memberikan kekuatan dan hidayah-Nya kepada penulis sehingga penulis mendapatkan kelancaran dalam menyelesaikan tugas akhir ini yang merupakan salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih penulis sampaikan kepada pihak-pihak yang telah mendukung, memberikan saran, motivasi, semangat, dan bantuan baik berupa materiil maupun moril demi tercapainya tujuan pembuatan tugas akhir ini. Tugas akhir ini tidak akan pernah terwujud tanpa bantuan dan dukungan dari berbagai pihak yang sudah melauangkan waktu, tenaga dan pikirannya. Secara khusus penulis akan menyampaikan ucapan terima kasih yang sebanyak-banyaknya kepada:

1. Kedua orang tua, kakak, adik dan seluruh keluarga yang selalu hadir dan senantiasa mendoakan dan memberikan kasih sayang serta semangat tiada henti untuk menyelesaikan Tugas Akhir ini.
2. Dosen pembimbing, Ibu Nur Aini Rakhmawati, S.Kom, M.Sc.Eng, Ph.D yang dengan sabar membimbing penulis untuk menyelesaikan tugas akhir ini.
3. Dosen Penguji, Bapak Radityo Prasetianto Wibowo, S. Kom., M. Kom , Ibu Irmasari Hafidz, S.Kom, M.Sc pada sidang proposal dan Faizal Johan Atletiko, S.Kom, M.T , Renny Pradina Kusumawardani, S.T, M.T, SCJP pada sidang tugas akhir yang telah memberikan banyak saran dan perbaikan dalam pengerjaan tugas akhir ini.
4. Bapak Nisfu Asrul Sani, S.Kom, M.Sc, selaku dosen wali penulis yang telah memberikan banyak saran dan

masukan selama penulis berkuliah di Departemen Sistem Informasi ITS.

5. Teman-teman Lab. ADDI dan MSI khususnya anak Egovbench periode 118 yang bersama-sama berjuang menyelesaikan tugas akhir di semester 8.
6. Dulur-dulur S.Pes Family yang selalu meluangkan waktunya untuk bertukar pendapat dan berbagi informasi lewat secangkir kopi panas di malam hari.
7. Manusia-manusia BegundalHood yang tidak pernah lelah memberikan informasi-informasi "positif" selama perkuliahan.
8. Teman-teman Osiris dan semua pihak yang terlibat dan membantu dalam pengerjaan tugas akhir ini yang belum mampu penulis sebutkan di atas.

Penyusunan laporan ini masih jauh dari kata sempurna sehingga penulis menerima adanya kritik maupun saran yang membangun untuk perbaikan di masa yang akan datang. Semoga buku tugas akhir ini dapat memberikan manfaat bagi pembaca.

Surabaya, 16 Juli 2018

Penulis,

Muhammad Fikry Hazmi

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR KODE

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab pendahuluan ini menguraikan proses identifikasi masalah penelitian yang meliputi latar belakang masalah, perumusan masalah, batasan masalah, tujuan tugas akhir, manfaat kegiatan tugas akhir dan relevansi terhadap pengerjaan tugas akhir. Berdasarkan uraian pada bab ini, harapannya gambaran umum permasalahan dan pemecahan masalah pada tugas akhir dapat dipahami.

1.1 Latar Belakang Masalah

Dengan berkembangnya teknologi informasi dan komunikasi (TIK), tidak dapat diingkari bahwa sedikit banyak telah mempengaruhi kehidupan di masyarakat. Hal tersebut mendorong pemerintah untuk menerapkan *e-government*, yaitu salah satu upaya pemerintah dalam memanfaatkan teknologi sebagai wadah untuk menyampaikan informasi yang cepat dan akurat. Penerapan *e-government* juga dapat meningkatkan pelayanan dengan efektif dan efisien. Selain itu, pemerintah dapat memberikan transparansi dan akuntabilitas dalam penyelenggaraan pemerintahan [1]. Media sosial merupakan salah satu penerapan dari *e-government* yang dipilih sebagai jembatan komunikasi antara masyarakat dan pemerintah.

Menurut *Digital Global Statistic* "Digital in 2018" [2], terdapat 130 juta jiwa aktif media sosial dari total populasi sebesar 265,4 juta jiwa di Indonesia. Angka tersebut meningkat 23% dari tahun sebelumnya. Dari total pengguna aktif media sosial di Indonesia, sebanyak 120 juta menggunakan *mobile phone* sebagai akses ke media sosial yang juga naik sebesar 30%. Dengan melihat peningkatan pengguna media sosial yang cukup tinggi pada masyarakat Indonesia, hal ini menjadi peluang bagi pemerintah untuk memanfaatkan media sosial dengan melibatkan masyarakat guna membangun pemerintahan yang lebih baik.

Dalam upaya untuk meningkatkan partisipasi publik, berbagai cara dapat dilakukan oleh pemerintah daerah. Salah satunya adalah dengan membangun komunikasi dua arah yang efektif dengan masyarakat melalui media sosial [3]. Komentar dari masyarakat dalam media sosial merupakan salah satu bentuk keterlibatan masyarakat terhadap pemerintahan. Dengan mengetahui topik pembicaraan masyarakat di media sosial pemerintahan, pemerintah dapat mengetahui tanggapan masyarakat terhadap kinerja yang telah dilakukan.

Pada penelitian ini, penulis akan melakukan klasifikasi komentar pada akun social media pemerintahan menggunakan algoritma Random Forest. Random Forest dipilih sebagai metode klasifikasi karena data yang akan digunakan besar dan metode ini dapat mengurangi kecenderungan kondisi *overfit* dalam klasifikasi. Hasil akhir dari penelitian ini bentuk visual dari hasil klasifikasi untuk mempermudah pemerintah dalam mengetahui topik yang banyak dibicarakan oleh netizen kepada pemerintah.

1.2 Perumusan Masalah

Berdasarkan uraian latar belakang, maka rumusan permasalahan yang menjadi fokus dan akan diselesaikan dalam tugas akhir ini antara lain:

1. Bagaimana cara melakukan akuisisi data komentar seluruh situs media sosial pemerintah daerah yang ada di Indonesia?
2. Bagaimana cara melakukan klasifikasi komentar pada media sosial pemerintah daerah?
3. Bagaimana cara merancang *platform* yang mampu melakukan visualisasi data hasil dari pengklasifikasian komentar?

1.3 Batasan Masalah

Batasan pemasalahan dalam pengerjaan tugas akhir ini adalah:

1. Studi kasus yang digunakan pada penelitian ini adalah media sosial Facebook, Twitter dan Youtube pemerintah daerah di Indonesia.
2. Data yang digunakan pada pengerjaan Tugas Akhir ini hanya dari data komentar/balasan pada halaman resmi Facebook, Twitter dan Youtube.
3. Pengambilan data menggunakan *API* setiap media sosial sesuai dengan kebijakan yang diberikan seperti limitasi dan sebagainya.
4. Penelitian ini menggunakan data komentar media sosial berbahasa Indonesia.
5. Proses pelabelan data anya dilakukan oleh satu annotator.
6. Proses *streaming* data menggunakan *mini-batching process* dan mengambil 10 *post* terbaru dari media sosial Facebook dan Youtube.
7. Klasifikasi komentar menggunakan algoritma Random Forest digunakan hanya sebatas untuk visualisasi dan mengetahui topik pembicaraan netizen pada sosial media pemerintahan.

1.4 Tujuan Tugas Akhir

Berdasarkan hasil perumusan masalah dan batasan masalah yang telah disebutkan sebelumnya, maka tujuan yang akan dicapai dari tugas akhir ini adalah untuk menyediakan perangkat lunak yang dapat melakukan pengklafisikasian komentar netizen pada media sosial pemerintahan guna memberikan informasi kepada pemerintah mengenai topik pembicaraan netizen.

1.5 Manfaat Tugas Akhir

Manfaat yang diharapkan dapat diperoleh dari tugas akhir ini adalah:

1. Mempermudah pemerintah untuk mengetahui apa dan seberapa besar topik pembicaraan netizen terhadap kinerja pemerintah.
2. Menyediakan data yang dapat digunakan untuk melakukan pengambilan keputusan dalam meningkatkan layanan pemerintah sesuai dengan kebutuhan di masyarakat.
3. Sebagai pembelajaran untuk penelitian di bidang *e-government* dan kaitannya dengan penggunaan algoritma *random forest* untuk melakukan klasifikasi topik yang banyak dibicarakan oleh netizen sosial media pemerintahan di Indonesia.

1.6 Relevansi

Relevansi tugas akhir ini terhadap laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) adalah karena tugas akhir ini berkaitan dengan mata kuliah Pemrograman Berbasis Web, Analisa dan Desain Perangkat Lunak dan Konstruksi Pengembangan Perangkat Lunak, Sistem Cerdas dan Penggalian Data Analitika Bisnis

BAB II
TINJAUAN PUSTAKA

Bab ini akan membahas penelitian sebelumnya yang berhubungan dengan tugas akhir dan teori - teori yang berkaitan dengan permasalahan tugas akhir ini.

2.1 Studi Sebelumnya

Pada subbab ini dijelaskan tentang referensi penelitian yang berkaitan dengan tugas akhir. Pada bagian ini memaparkan acuan penelitian sebelumnya yang digunakan oleh penulis dalam melakukan penelitiannya.

1. Penelitian pertama adalah *Rancang Bangun Perangkat Lunak Benchmarking Sosial Media Pemerintah Daerah Indonesia*, oleh Abi Nubli Abadi [4]. Pada penelitian ini dilakukan pembuatan metode dan aplikasi penilaian terhadap media sosial pemerintah daerah Indonesia dengan menggunakan metode *Web-Crawling*, *TF-IDF* dan *Regular Expression* serta menggunakan teknologi *MySQL*. Luarannya adalah visualisasi peringkat dari penilaian seluruh media sosial pemerintah daerah Indonesia.
2. Penelitian kedua adalah *Facebook Usage in Government – A Case Study of Information Content*, oleh Monika Magnusson, Peter Bellström, dan Claes Thorén [5]. Tujuan penelitian ini adalah ingin mengetahui informasi apa yang diberikan netizen kepada pemerintah melalui kiriman pada halaman media sosial *pemerintahan*. Penulis melakukan kategorisasi dalam menilai partisipasi online pengguna media sosial dengan mengkategorisasikan kiriman netizen ke dalam 11 kategori yaitu *Citizen Marketing Events, Citizen Sharing Information, Complaining about Municipality, Express Opinion, Identity or Community Building, Marketing of*

Business, Praising Municipality, Request of Existing Service, Request of Future Service, Request for Information, dan *Report Service Breakdown*. Tujuan adanya Facebook mengundang netizen untuk melaporkan kerusakan layanan, meminta layanan yang kurang, dan membantu kinerja pemerintah dengan berdiskusi rutin dan masalah praktis. Diskusi yang terjadi antar pengguna maupun pengguna dengan pemerintah pun dianalisis untuk mendapatkan pemahaman tentang apa yang sedang dibahas. Hasil analisis tersebut dapat dijadikan acuan dan strategi untuk meningkatkan pelayanan kepada masyarakat.

3. Penelitian ketiga adalah *A Systematic Analysis of Random Forest Based Social Media Spam Classification,* oleh Mohammed Al-Janabi dan Peter Andras [6]. Pada paper ini, dilakukan pengklafikasian *spam* pada *tweet* di media sosial yang berisi *malicious URL* dengan menggunakan metode *Random Forest.* Dataset yang digunakan untuk *training* dalam penelitan ini adalah *tweet* yang berisi URL. Ada sebanyak 39017 *tweet* yang digunakan, termasuk 18075 yang positif *spam* dan 20942 negatif *spam*. Rata-rata mesin mendeteksi *tweet spam* dengan URL jahat memberikan nilai *F1-measure* sebesar 0,885, dengan *precision* sebesar 96% dan *recall* sebesar 82,2%. Terdapat 3 paramater yang digunakan pada penelitian ini yaitu banyak jumlah *trees,* maksimum *tree depth,* dan ukuran minimum *leaf nodes.* Pengaruh kenaikan jumlah *trees* terlihat dengan maksimum *tree depth* yang lebih besar. Rata-rata akurasi tidak meningkat secara signifikan setelah menggunakan lebih dari 9 *trees.* Pengaruh kenaikan maksimum *tree depth* terlihat signifikan sampai maksimum *tree depth* 16 dengan ukuran minimum *leaf nodes* yang besar dan sampai maksimum *tree depth* 24 dengan ukuran minimum *leaf nodes* yang kecil.

4. Penelitian keempat adalah *A Machine Learning Analysis of Twitter Sentiment to the Sandy Hook Shootings,* oleh

Nan Wang, Blesson Varghese dan Peter D. Donnelly [7]. Pada paper ini, dilakukan analisa sentimen terhadap *pro-gun* dan *anti-gun* yang dinyatakan di media sosial *twitter,* sebagai tanggapan atas penembakan di Sandy Hook Basic School pada tahun 2012. Dataset yang digunakan pada penelitian ini adalah kumpulan *tweet* yang berisi satu atau lebih kata kunci yang telah ditentukan sebelumnya. Beberapa algoritma dipakai untuk menentukan perbandingan hasilnya yaitu SVM, *Maximum Entropy* (ME), *Decision Tree*, *Bagged Tree*, *Boosted Tree*, *Random Forest* (RF), *Neural Network* (NN) and *Naıve Bayes* (NB). Peneliti membandingkan kinerja model yang dihasilkan oleh ukuran data pelatihan yang berbeda antara 1000, 2000 sampai 5000 *tweets* dan algoritma yang berbeda menggunakan fitur *uni-gram*. Model terbaik dari hasil *training* adalah RF dengan akurasi hampir 92%, diikuti oleh *Bagged Tree* dan *Boosted Tree* yang memiliki kinerja hampir serupa dan SVM yang memiliki akurasi hampir 85%.

2.2 Dasar Teori

Berisi teori-teori yang mendukung serta berkaitan dengan tugas akhir yang sedang dikerjakan.

2.2.1 *E-Government*

E-government atau pemerintahan elektronik adalah sebuah gagasan yang diajukan oleh mantan wakil presiden AS (Al Gore), dalam visinya untuk menghubungkan masyarakat ke berbagai instansi pemerintah. Hal ini dilakukan untuk mendapatkan semua jenis layanan pemerintah secara otomatis, sehingga dalam menyelesaikan pekerjaannya pemerintah bergantung pada informasi dan jaringan komunikasi untuk mengurangi biaya, meningkatkan kinerja, kecepatan penyampaian dan efektivitas implementasinya [8].

Bank Dunia, (2001) mendefinisikan *E-government* sebagai sistem informasi dan komunikasi yang dimiliki atau dioperasikan pemerintah yang mengubah hubungan dengan

warga negara, sektor swasta dan/atau lembaga pemerintah lainnya sehingga dapat mendorong pemberdayaan masyarakat, memperbaiki pemberian layanan, memperkuat akuntabilitas, meningkatkan transparansi, atau meningkatkan efisiensi pemerintah [9].

Berdasarkan pengertian-pengertian di atas dapat disimpulkan bahwa *E-Government* merupakan pemanfaatan teknologi informasi oleh institusi pemerintahan untuk meningkatkan kinerja pemerintah dalam hubungan dengan masyarakat dan pihak lain yang terkait agar menjadi lebih efektif dan efisien.

2.2.2 Media Sosial

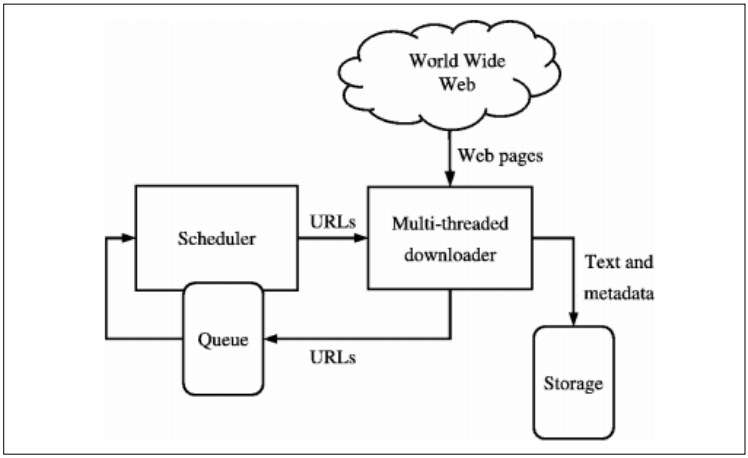
Media sosial merupakan sebuah media yang menggunakan internet dan memungkinkan penggunanya bisa dengan mudah untuk berpartisipasi, saling berbagi dan menciptakan isi meliputi blog, wiki, jejaring sosial, konten virtual dan situs berbagi media [10]. Media sosial mendorong perubahan sosial dengan memberikan masyarakat tempat untuk mengekspresikan pemikiran dan opini mereka dan dibagikan dengan orang lain. Seorang individu menyadari bahwa mereka tidak berbicara sendiri, orang lain ikut berpatisipasi dengan sangat responsif dan ikut ambil bagian dalam percakapan, serta melihat sudut pandang dan mendengar suara mereka. Hal ini memulai pergeseran sosial menuju kekuatan yang dikendalikan oleh massa [11]. Media sosial terbesar antara lain Facebook, Twitter, dan Youtube. Media sosial yang dimiliki oleh pemerintahan di Indonesia terdapat total sebanyak 110 akun Facebook, 92 akun Twitter, dan 64 akun Youtube [12].

Berdasarkan Permenpan No. 83 tahun 2012 tentang Pedoman Pemanfaatan Media Sosial Instansi Pemerintahan, pengukuran yang digunakan untuk mengukur pengaruh dampak media sosial adalah kekerapan diskusi mengenai isi atau pesan tertentu yang disampaikan, komentar, dan efek penyebarluasan informasi (komunikasi viral) [13]. Dengan adanya media sosial, pemerintah mengajak siapa saja yang tertarik untuk saling berpartisipasi dengan memberi kontribusi dan *feedback* secara

terbuka, memberi komentar, serta berbagi informasi dalam waktu cepat dan tak terbatas.

2.2.3 *Web Crawler*

Web crawler adalah komponen penting dari mesin pencari web, di mana mereka digunakan untuk mengumpulkan korpus halaman web yang diindeks oleh mesin pencari [14]. *Web crawler* (juga disebut sebagai *robot* atau *spider*) adalah sistem untuk mengunduh halaman web secara massal dan digunakan untuk berbagai tujuan. Sistem ini merupakan salah satu komponen utama mesin pencari web yang merakit korpus halaman web, mengindeksnya, dan memungkinkan pengguna mengeluarkan query terhadap indeks dan menemukan halaman web yang sesuai dengan query [15]. Arsitektur dari *web crawler* sesuai dengan Gambar 2.2.1.



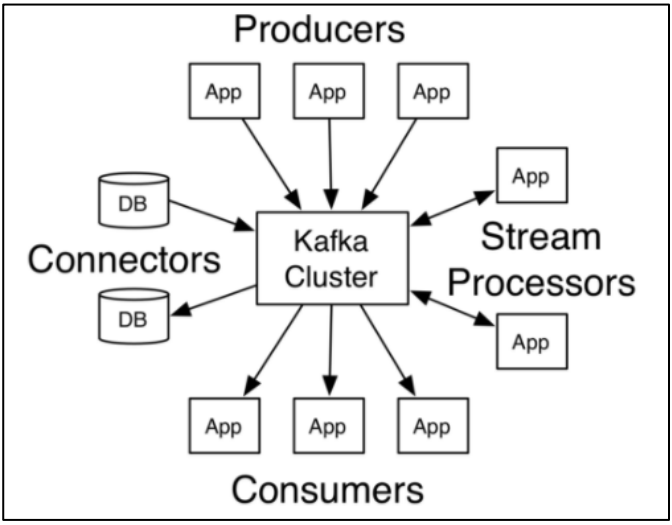
Gambar 2.2.1 Arsitektur *Web Crawler*

Berdasarkan Gambar 2.2.1, cara kerja dari *Web Crawler* untuk mendapatkan informasi yaitu dimulai dari sebuah *World Wide Web* (WWW) atau halaman *web*, kemudian menggunakan *multi-threaded downloader* untuk mengunduh data yang dibutuhkan. Data dapat dijadwalkan sesuai dengan kebutuhan dengan bantuan s*cheduler* dan *URL Queue*. *Output* berupa t*ext*

dan *metadata* yang sudah di-*crawl* akan disimpan dalam *storage* [16].

2.2.4 *Kafka*

Kafka merupakan *platform streaming* terdistribusi yang digunakan untuk *publish-subscribe* pada *stream* beberapa *record*. *Kafka* mereplikasi partisi log topik ke beberapa server dan dirancang untuk memungkinkan aplikasi kita memproses beberapa *records* saat itu juga. Platform ini menggunakan IO secara efisien dengan sistem *batching* dan melakukan *compress* terhadap *records* membuat *Kafka* begitu cepat. Selain itu, platform ini juga digunakan untuk memisahkan *data stream*, mengalirkan data ke *data lakes*, aplikasi dan sistem analisis arus *real-time* [17].



Gambar 2.2.2 Arsitektur Kafka

Berdasarkan Gambar 2.2.2 *Kafka* mempunyai 4 API inti pada prosesnya yaitu *Producers*, *Consumers*, *Streams*, dan *Connectors*. *Producers* memungkinkan aplikasi mempublikasikan aliran *record* ke satu atau beberapa topik *Kafka*. *Consumers* memungkinkan aplikasi *subscribe* ke satu atau beberapa topik dan memproses arus *records* yang

dihasilkannya. *Streams* memungkinkan aplikasi bertindak sebagai *stream processor*, yang menggunakan *input stream* dari satu topik atau lebih dan menghasilkan *output stream* ke satu atau lebih *output stream*, yang secara efektif mengubah *input stream* ke *output stream*. Sedangkan *Connectors* memungkinkan membangun dan menjalankan *producer* atau *consumer* yang dapat digunakan kembali yang menghubungkan topik *Kafka* dengan aplikasi atau sistem data yang ada. Sebagai contoh, sebuah konektor ke database relasional dapat menangkap setiap perubahan ke sebuah tabel [18].

2.2.5 Klasifikasi

Klasifikasi adalah teknik penggalian data (teknik mesin) yang digunakan untuk memprediksi keanggotaan grup pada setiap kasus dalam data [19]. Klasifikasi merupakan *supervised learning* dimana setiap *instance* dimiliki oleh sebuah *class*, yang ditunjukkan oleh nilai dari atribut sasaran khusus atau hanya atribut kelas. Atribut terdiri dari dua bagian, yaitu atribut prediktor dan atribut tujuan. Atribut prediktor digunakan untuk memprediksi nilai yang terakhir dan harus relevan untuk memprediksi kelas *instance.*

Dalam klasifikasi, data yang digunakan dibagi menjadi dua bagian yaitu *training set* dan *test set* [20]. Kemudian proses klasifikasi juga dibagi menjadi dua, yaitu tahap *training* dan *testing*. Pada tahap *training*, algoritma memiliki akses terhadap nilai atribut prediktor dan atribut tujuan untuk semua contoh rangkaian pelatihan, dan menggunakan informasi tersebut untuk membangun model klasifikasi. Model ini mewakili pengetahuan klasifikasi. Pada tahap *testing*, *class* pada *test set* tidak diketahui dan dilakukan prediksi *class* berdasarkan model yang dibuat pada tahap sebelumnya. Tujuan utama algoritma klasifikasi adalah untuk memaksimalkan akurasi prediktif yang diperoleh dari model klasifikasi pada tahap *training* [20].

Klasifikasi komentar netizen merupakan klasifikasi konten pada komentar yang dilakukan oleh pemda. Klasifikasi ini diambil berdasarkan pada paper penelitian kedua [5]. Dataset

yang digunakan pada penelitian tersebut adalah data komentar pada halaman resmi Facebook milik pemerintahan Swedia. Daftar kategori beserta deskripsi dapat dilihat pada Tabel 2.2.1.

Tabel 2.2.1 Klasifikasi Komentar

Kategori	Deskripsi
Berbagi Informasi	Berbagi Informasi, bukan suatu acara atau bisnis
Meminta Informasi	Pertanyaan kepada pemerintah atau netizen lainnya
Mengemukakan Pendapat	Memberikan sudut pandang dalam masalah pemerintahan
Melaporkan Permasalahan Layanan	Laporan Kerusakan layanan pemerintah
Apresiasi	Memberikan Feedback Positif
Komplain	Masyarakat memegang tanggung jawab pemerintahan
Meminta Peningkatan Layanan	Permintaan pengingkatan layanan yang sudah ada
Meminta Layanan Baru	Permintaan layanan baru
Memasarkan Acara	Promosi acara non-komersil
Identitas atau Pengenalan Komunitas	Promosi diri atau pemerintahan

2.2.6 *Random Forest Algorithm*

Random Forest adalah classifier berbasis *ensemble*, dimana terdiri dari koleksi sub-model yang digunakan untuk membuat keputusan bersama. *Random forest* menggunakan sejumlah klasifikasi *Decision Tree* atau pohon keputusan. Pohon-pohon ini dibangun dengan menggunakan sampel acak dari kumpulan

data pelatihan lengkap, yang menghasilkan perbedaan potensial antar setiap *tree*, karena peringkat kepentingan fitur berbeda untuk *tree* yang berbeda. Setiap pohon dilatih pada contoh data pelatihan *bootstrap* dan pada setiap *node* algoritma hanya mencari di subset acak dari variabel untuk menentukan *split*. *Bootstrapping* adalah teknik umum yang secara iteratif melatih data pelatihan dan mengevaluasi pengklasifikasi untuk meningkatkan kinerjanya [21]. Ketergantungan pada beberapa pohon keputusan membuat *classifier random forest* lebih kuat dan lebih sedikit rentan terhadap *overfitting* dibandingkan dengan pohon keputusan tunggal dan metode *non-ensemble* lainnya [22].

Membangun *tree* membutuhkan identifikasi beberapa parameter, seperti jumlah *features* yang digunakan, *tree depth*, dan minimum *leaf size* [23]. Jumlah *features* diberikan oleh dimensi data yang digunakan. Maksimum *tree depth* adalah jumlah maksimum keputusan biner berturut-turut yang harus dimiliki oleh *decision tree*. Minimum *leaf size* adalah jumlah minimum item data yang diharapkan termasuk dalam subkumpulan data yang terkait dengan *leaf node* pohon keputusan. Jika *splitting* dari subset yang terkait dengan *leaf node* menghasilkan *leaf node* yang memiliki lebih sedikit item data yang terkait dengannya dari minimum *leaf size*, *splitting* tidak terjadi dan *node* tetap sebagai *leaf node*.

Berikut merupakan langkah-langkah dari algoritma *random forest* [24]:

N *trees* menjadi jumlah *tree* yang akan dibangun untuk setiap pohon N iterasi.

1. Pilih *sample bootstrap* baru dari set pelatihan.
2. Membangun *un-pruned tree* pada *bootstrap* tersebut.
3. Pada setiap *node* dalam, memilih secara acak *mtry* dan menentukan *split* terbaik hanya dengan menggunakan prediktor ini.

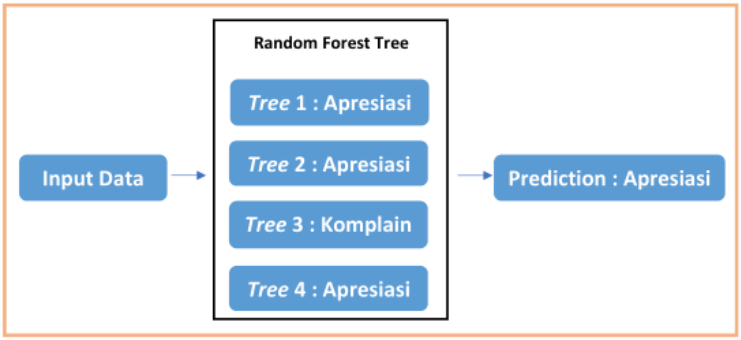
4. Hasil keseluruhan prediksi sebagai mayoritas pilihan dari semua pohon yang dilatih secara individual.

Contoh penggunaan klasifikasi menggunakan *Random Forest* akan dijelaskan sebagai berikut.

Tabel 2.2.2 Contoh klasifikasi menggunakan *Random Forest*

	Dok	Kata-kata	Kelas
Train	1	layan baik ramah	Apresiasi
	2	kerja cepat baik tanggap	Apresiasi
	3	tunggu lama emosi	Komplain
	4	layan cepat tanggap	Apresiasi
	5	lambat tidak profesional	Komplain
Test	6	baik ramah tidak lama tanggap	???

Dari Tabel 2.2.2 di atas, dapat dilihat bahwa data terdiri dari data *training* dan data *testing*. Dari data *training* tersebut dibangun menjadi beberapa *tree* dan nantinya *input data* yang ingin dilaukan prediksi kelas terhadap semua *tree*, contoh prediksi tiap-tiap *tree* dapat dilihat pada Gambar 2.2.3 Cara kerja *Random Forest*.



Gambar 2.2.3 Cara kerja *Random Forest*

Hasil prediksi akhir didapatkan dari *voting* terbanyak dari hasil tiap-tiap *tree* yang dibangun, untuk contoh studi kasus ini adalah data tersebut memiliki kelas Apresiasi.

2.2.7 *Precision, Recall,* dan *F-Measure*

Precision, Recall, dan *F-Measure* merupakan parameter yang digunakan dalam pengujian hasil klasifikasi. *Precision* adalah proporsi kasus Prediksi Positif yang benar-benar Positif. Sedangkan *Recall* adalah proporsi kasus *Real Positive* yang diprediksi positif dengan benar [25]. Untuk ilustrasinya, setiap objek dikaitkan dengan *Predicted Class* sebagai nilai prediksi dan *Actual Class* sebagai nilai sebenarnya. Hasil eksperimen dirangkum dalam c*onfusion matrix* pada Tabel 2.2.3 [26]:

Tabel 2.2.3 *Confusion matrix*

		Actual class	
		+	-
Predicted class	+	TP	FN
	-	FP	TN

Secara umum, *precision* dan *recall* dapat dirumuskan sesuai dengan persamaan (1) dan (2):

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Dari dua rumus tersebut, yang dimaksud dengan TP, TN, FP, dan FN adalah:

- *True Positives* (TP), data yang diklasifikasikan oleh model sebagai positif dan label sesungguhnya adalah memang positif (prediksi benar).

- *True Negatives* (TN), data yang diklasifikasikan oleh model sebagai negatif dan label sesungguhnya adalah memang negatif (prediksi benar).
- *False Positives* (FP), data yang diklasifikasikan oleh model sebagai positif dan label sesungguhnya adalah negatif (prediksi salah).
- *False Negative* (FN), data yang diklasifikasikan oleh model sebagai negatif dan label sesungguhnya adalah positif (prediksi salah).

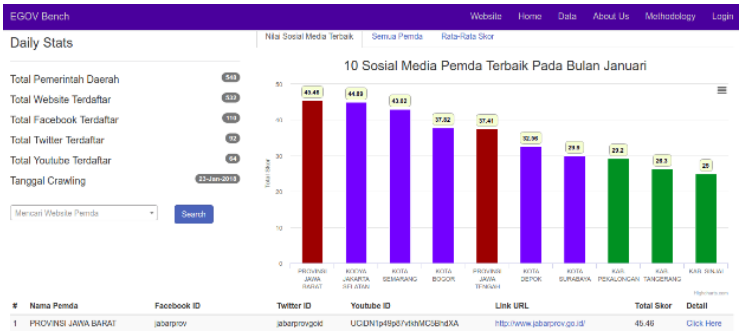
Untuk mengoptimalkan hasil yang didapatkan, diperlukan parameter pengukuran lainnya yaitu *F-measure*. *F-measure* secara efektif mereferensikan *True Positive* ke rata-rata dari Prediksi Positif dan *Real Positive* [25]. *F-measure* dapat dirumuskan sesuai dengan persamaan (3):

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

Semakin besar nilai *F-measure*, maka semakin baik pula hasil dari klasifikasi tersebut.

2.2.8 *Egovbench*

Egovbench adalah aplikasi untuk melakukan pemantauan *website* pemerintah daerah (provinsi, kabupaten, dan kota) yang ada di Indonesia. Dengan adanya aplikasi ini diharapkan dapat meningkatkan kesadaran pemerintah untuk meningkatkan mutu dan kualitas dari *e-government* milik pemerintah tersebut agar terus menjadi lebih baik, sehingga masyarakat dan pemerintah dapat melihat perkembangan dari *e-government* masing-masing daerahnya [12]. Antarmuka dari *website Egovbench* ini dapat dilihat pada Gambar 2.2.4.



Gambar 2.2.4 ***Interface web EGov Benchmark***

Fitur yang dihadirkan oleh aplikasi ini berupa penilaian *website* dan media sosial yang dimiliki masing-masing pemerenitah daerah. Analisa mencakup seluruh pemerintah daerah Indonesia dan dilakukan berdasarkan waktu. Aplikasi ini dibangun menggunakan bahasa pemrograman *web*, yaitu *PHP* untuk keseluruhan *website* dan *Phyton* untuk tahapan praproses data sebelum divisualisasikan.

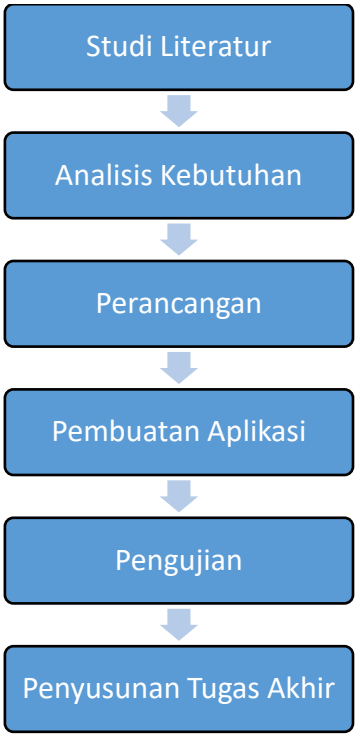
Halaman ini sengaja dikosongkan

BAB III
METODOLOGI

Bab ini menjelaskan tentang metodologi yang akan digunakan dalam penyusunan tugas akhir. Metodologi akan digunakan sebagai panduan dalam penyusunan tugas akhir agar terarah dan sistematis.

3.1 Tahapan Pelaksanaan Tugas Akhir

Pada subbab ini akan menjelaskan mengenai metodologi dalam pelaksanaan tugas akhir. Metodologi ini dapat dilihat pada Gambar 3.1.1.



Gambar 3.1.1 Metodologi tugas akhir

3.2 Uraian Metodologi

Pada bagian ini akan dijelaskan secara lebih rinci masing-masing tahapan yang dilakukan untuk penyelesaian tugas akhir ini.

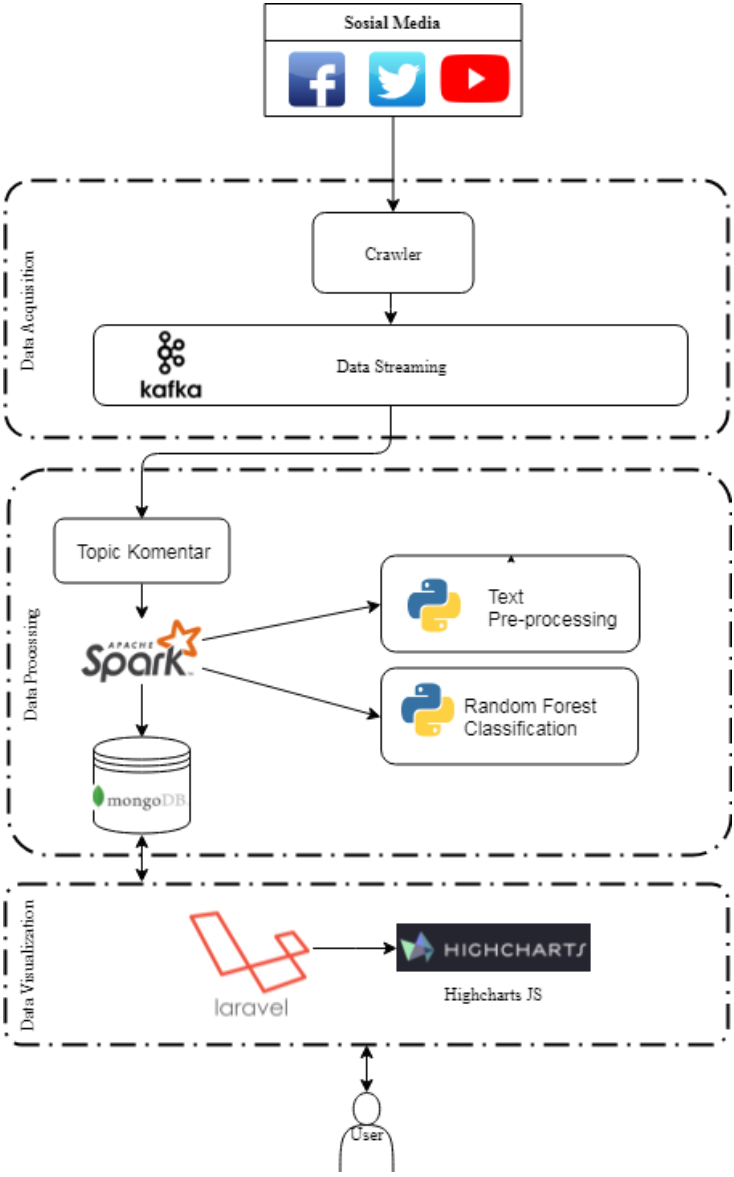
3.2.1 Studi Literatur

Tahap studi literatur ini dilakukan untuk memahami konsep, metode, dan teknologi sesuai bahasan dan permasalahan sehingga dapat memberikan solusi mengenai permasalahan yang akan digunakan dalam penyusunan tugas akhir. Adapun literatur yang digunakan dalam penelitian ini adalah terkait klasifikasi, *Random Forest*, dan evaluasi model mengacu pada penlitian dengan judul *Web document classification by keywords using random forests* [27], penelitian berjudul *Random Forest Classifier for Multi-Category Classification of Web Pages* [21], penelitian berjudul *A Systematic Analysis of Random Forest Based Social Media Spam Classification* [6]. Sedangkan kategori komentar mengacu pada penelitian berjudul *Facebook Usage in Government – A Case Study of Information Content* [5].

3.2.2 Analisis Kebutuhan

Pada tahap ini dilakukan penggalian kebutuhan pengguna untuk mempermudah pemahaman konteks bisnis dari aplikasi itu sendiri. Kebutuhan seperti dataset untuk pembuatan model klasifikasi beserta semua *library* yang akan digunakan dan kebutuhan pada prsoes *streaming* data untuk dilakukan pengklasifikasian. Selain itu pada tahap ini juga dilakukan pendefinisian *output* yang akan dihasilkan, fitur yang dimiliki oleh aplikasi, fungsi dari aplikasi yang dikembangkan.

3.2.3 Perancangan



Gambar 3.2.1 Arsitektur Sistem

Gambar 3.2.1 merupakan alur aplikasi yang akan dibangun pada penelitian ini. Pada tahap ini, hasil dari analisis kebutuhan digunakan untuk merencanakan cara kerja sistem dalam hal desain arsitektur, desain antarmuka, *database*, dan desain aplikasi. Berawal dari proses *streaming* data menggunkan *API* masing-masing media sosial akan di-*produce* di Kafka untuk setiap media sosial. Data diambil topik-topik yang berisikan komentar. Data tersebut akan dilakukan tahap praproses dan hasil dari praproses akan diklasifikasikan menggunakan Apache Spark. Penggunaan Apache Spark juga untuk membantu proses *streaming*. Proses *streaming* pada Apache Spark tidak sepenuhnya *real-time*, melainkan ada proses *micro-batching* pada data yang masuk. Data yang sudah diklasifikasikan akan dimasukkan kedalam database MongoDB dan akan divisualisasikan melalui web menggunakan *framework* Laravel dan Highcarts sebagai library untuk pembentukan grafik. Banyak data yang terklasifikasi pada setiap pemda dapat dilihat oleh semua pemda.

3.2.4 Pembuatan Aplikasi

Tahap ini adalah tahapan dimana rancangan yang telah dibuat pada tahap sebelumnya dikonversikan ke dalam bahasa pemrograman atau disebut pengkodean. Pada tahap ini, pengkodean terbagi menjadi tiga, yaitu:

1. Rancang bangun *Web Crawler*
   Web crawler mulai dikodifikasi untuk mengakuisisi data yang digunakan untuk pengklasifikasian. Dengan menggunakan *Kafka* sebagai *aggregator*, data yang diambil berupa komentar/*tweet mention* dari media sosial pemerintah. Untuk media sosial Facebook, data yang diambil hanya dari *official page*. Kemudian data disimpan dalam database dan dilakukan praproses pada data tersebut untuk membersihkan data yang akan digunakan, yaitu *data cleaning, case folding, tokenization, stemming*, dan *stopwords removal*.
2. Rancang bangun Pengklasifikasian

Kodifikasi pada pengklasifikasian berupa praproses semua data komentar yang sudah disimpan dan diberikan label yang sesuai. Proses *labeling* yang dilakukan dengan cara manual. Setelah itu, dilakukan pengklasifikasian menggunakan metode *Random Forest* dengan Bahasa pemrograman *Python* dan menggunakan *Apache Spark*.

3. Rancang bangun Web Visualisasi

   Data yang sudah melewati proses pengklasifikasian kemudian dibuat visualisasinya dalam bentuk grafik pada *website*. Grafik yang digunakan adalah diagram batang untuk menggambarkan data nominal dan menggambarkan perbandingan dari topik yang dibicarakan netizen. serta diagram garis untuk meunjukkan tren masuknya data komentar. Penggunaan grafik ini merujuk pada penelitian *ReVision: Automated Classification, Analysis and Redesign of Chart Images* [28]. Web visualisasi dibangun menggunakan bahasa pemrograman web *PHP* dan *framework* Laravel.

3.2.5 Pengujian

Pada tahap ini dilakukan pengujian terhadap aplikasi yang telah dibangun. Pengujian dilakukan untuk melihat apakah aplikasi telah berjalan sesuai dengan rancangan atau masih terdapat *error* dan *bug* sehingga masih perlu dilakukan perbaikan aplikasi. Sedangkan validasi model dilakukan dengan menggunakan parameter *precision*, *recall*, dan *F-measure* untuk 10 kategori berdasarkan penelitian yang dilakukan oleh Monika Magnusson, Peter Bellström, dan Claes Thorén [5] pada komentar netizen di media sosial pemerintahan.

3.2.6 Penyusunan Laporan Tugas Akhir

Tahapan terakhir adalah penyusunan laporan tugas akhir sebagai bentuk dokumentasi atas terlaksananya tugas akhir ini. Laporan tugas akhir dibuat sesuai dengan format yang telah ditentukan. Tahapan penyusunan laporan tugas akhir dilakukan

sejak awal hingga berakhirnya proses pengerjaan tugas akhir ini.

BAB IV
PERANCANGAN

Pada bab ini membahas terkait alur perancangan terkait beberapa hal yang diperlukan dalam proses pembuatan aplikasi sesuai dengan alur yang dijelaskan pada bab metodologi. Rancangan yang diuraikan meliputi subyek dan obyek penelitian seperti pemlihan subyek dan obyek penelitian serta bagaimana penelitian ini dilakukan.

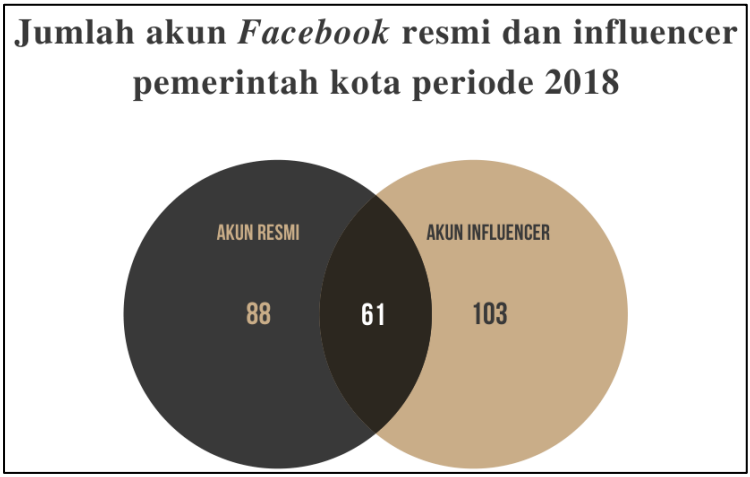
4.1 Akuisisi Data

Dalam melakukan analisis dan visualisasi klasifikasi komentar netizen pada media sosial pemerintahan di Indonesia, data berupa komentar merupakan obyek utama yang dianalisis. Namun dalam melakukan pengumpulan data, objek yang diambil merupakan *post* dan komentar pada sosial media Facebook, Twitter, dan Youtube pada akun resmi pemerintahan dan akun *influencer* di setiap daerah. Jumlah akun pada masing-masing media sosial pemerintahan dan *influencer* per April 2018 dapat dilihat pada Tabel 4.1.1.

Tabel 4.1.1 Jumlah Akun Media Sosial Periode April 2018

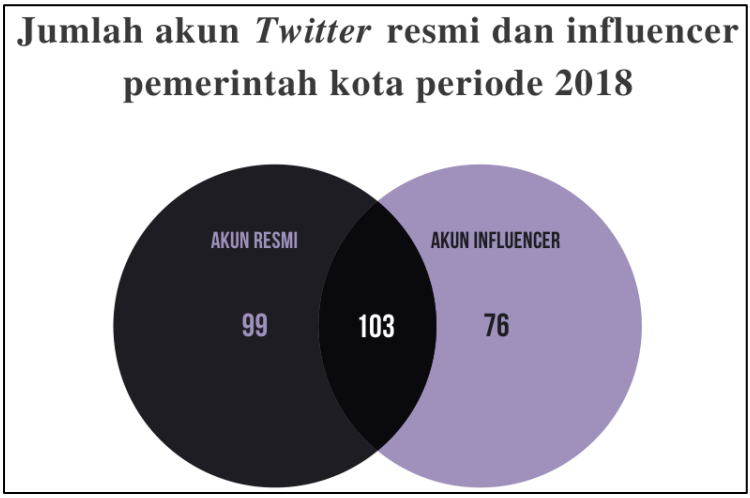
Media Sosial	Jumlah Akun Resmi Pemerintahan	Jumlah Akun Influencer
Facebook	149	164
Twitter	202	179
Youtube	113	60

Pada media sosial Facebook, terdapat 88 pemerintahan yang memiliki akun resmi, 103 pemerintahan yang memiliki akun *influencer*, 61 pemerintahan yang keduanya memiliki akun resmi dan *influencer*. Dapat dilihat melalui diagram venn pada Gambar 4.1.1.



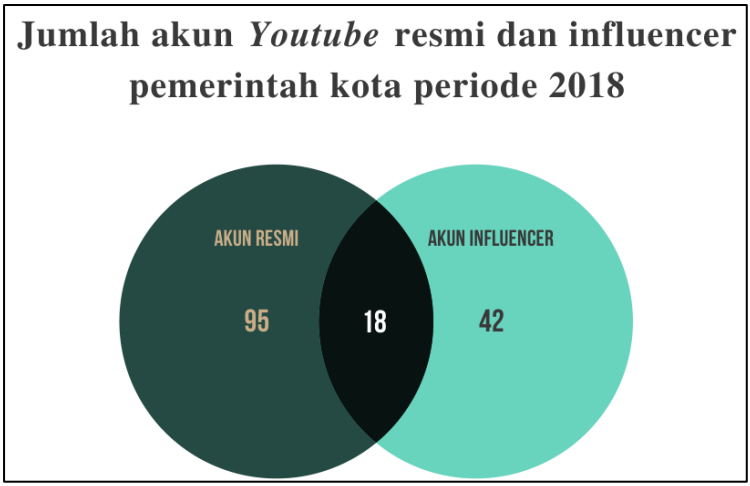
Gambar 4.1.1 Jumlah akun *Facebook*

Pada media sosial Twitter, terdapat 99 pemerintahan yang memiliki akun resmi, 76 pemerintahan yang memiliki akun *influencer*, 103 pemerintahan yang keduanya memiliki akun resmi dan *influencer*. Dapat dilihat melalui diagram venn pada Gambar 4.1.2.



Gambar 4.1.2 Jumlah akun *Twitter*

Pada media sosial Youtube, terdapat 95 pemerintahan yang memiliki akun resmi, 42 pemerintahan yang memiliki akun *influencer*, 18 pemerintahan yang keduanya memiliki akun resmi dan *influencer*. Dapat dilihat melalui diagram venn pada Gambar 4.1.3.



Gambar 4.1.3 Jumlah akun *Youtube*

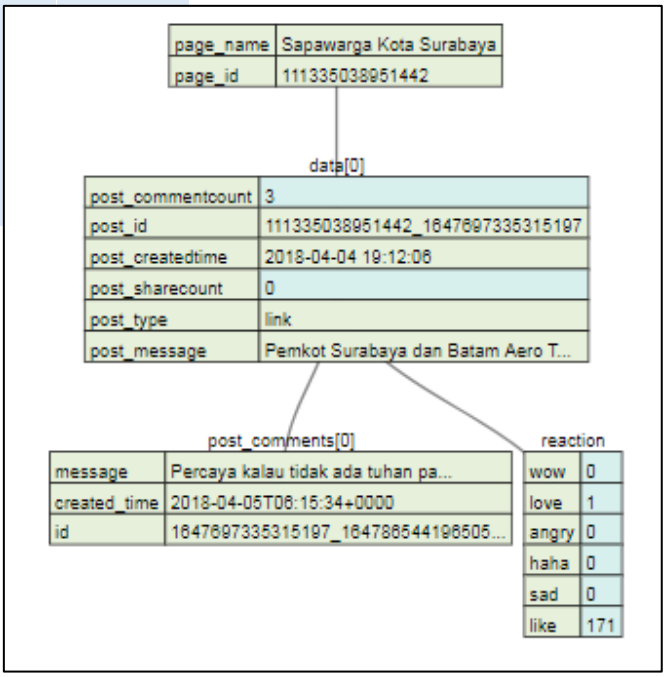
Proses pengambilan data dilakukan dengan dua cara yaitu *crawling* dan *streaming*. Data yang diambil dengan cara *crawling* digunakan sebagai bahan dalam pembuatan model, sedangkan data yang diambil dengan *streaming* digunakan sebagai data yang nantinya akan diklasifikasikan. Proses pengambilan data menggunakan API yang disediakan pada masing-masing media sosial. Adapun atribut data yang diakuisisi pada media sosial Facebook dapat dilihat pada Tabel 4.1.2 dan untuk lebih jelasnya digambarkan tree diagram pada Gambar 4.1.4.

Tabel 4.1.2 JSON Facebook

Object	Keterangan	Contoh
page_id	**Nomor ID facebook pemda**	**111335038951442**

page_name			**Nama facebook pemda**	**Sapawarga Kota Surabaya**
posts	post_id		Nomor ID postingan	111335038951442_1647697335315197
	post_message		Isi postingan	Pemkot Surabaya dan Batam Aero Technic (BAT) Siap Jalin Kerjasama.
	post_type		Tipe post yang dikirim	link
	post_sharecount		Jumlah tautan dibagikan	0
	post_comment	**id**	**Nomor ID komentar**	**1647697335315197_1647865441965053**
		message	**Isi komentar**	**Semoga surabaya menjadi kota yang lebih baik lagi**
		created_time	**Waktu pengiriman komentar**	**2018-04-05T10:51:17+0000**
	post_commentcount		Jumlah komentar	3
	reaction	like	Jumlah reaction like	171
		love	Jumlah reaction love	1
		wow	Jumlah reaction wow	0

		haha	Jumlah reaction haha	0
		sad	Jumlah reaction sad	0
		angry	Jumlah reaction angry	0
	created_time		Waktu pengiriman post	2018-04-04 19:12:06

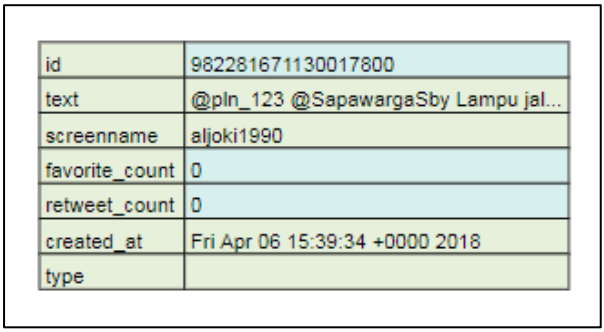


Gambar 4.1.4 Tree Diagram Facebook JSON

Sedangkan untuk atribut data yang diakuisisi pada media sosial Twitter dapat dilihat pada Tabel 4.1.3 dan untuk lebih jelasnya digambarkan tree diagram pada Gambar 4.1.5.

Tabel 4.1.3 JSON Twitter

Object	Keterangan	Contoh
id	**Nomor ID tweet**	**982281671130017793**
text	**Isi tweet**	**@pln_123 @SapawargaSby Lampu jalan kota di JL. TAMBANG BOYO mati... Kenapa yaa?? Seremm tau ga sih.. Tolong segera di nyalakan...**
screenname	Nama akun	aljoki1990
favorite_count	Jumlah favorit pada tweet	0
retweet_count	Jumlah retweet pada tweet	0
created_at	**Waktu pengiriman post**	**Fri Apr 06 15:39:34 +0000 2018**
type	Tipe media tweet	



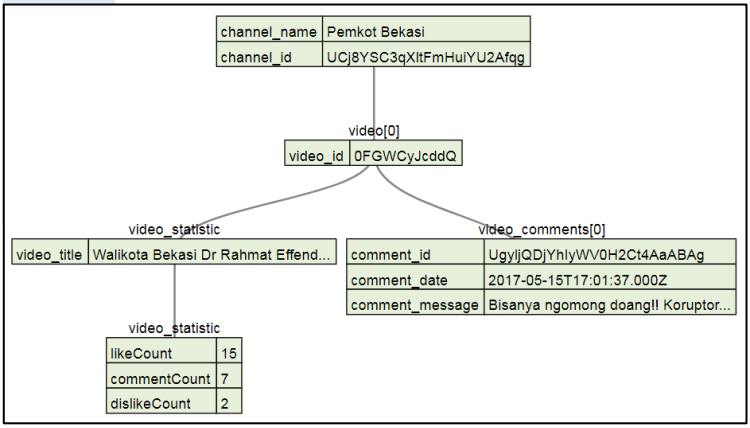
Gambar 4.1.5 Tree Diagram Twitter JSON

Untuk atribut data yang diakuisisi pada media sosial Youtube dapat dilihat pada Tabel 4.1.4 dan untuk lebih jelasnya digambarkan tree diagram pada Gambar 4.1.6.

Tabel 4.1.4 JSON Youtube

Object				Keterangan	Contoh
channel_id				Nomor ID channel	UCj8YSC3qXltFmHuiYU2Afqg
channel_name				Nama youtube pemda	Pemkot Bekasi
video	video_id			Nomor ID video	0FGWCyJcddQ
	video_info	video_title		Judul video	Walikota Bekasi Dr Rahmat Effendi : Explore Potensi Curug Parigi untuk Wujudkan #Bekasi Hijau
		video_statistic	likecount	Jumlah like video	15
			comment count	Jumlah kometar video	7
			dislikecount	Jumlah dislike video	2
	video_comments	**comment_id**		**Nomor ID komentar**	**UgyljQDjYhIyWV0H2**

				Ct4AaABAg
		comment_date	**Waktu pengiriman komentar**	**2017-05-15T17:01:37.000Z**
		comment_message	**Isi komentar**	**Bisanya ngomong doang**



Gambar 4.1.6 Diagram Tree Youtube JSON

4.2 Desain Database

Database yang digunakan pada aplikasi untuk melakukan *stream* data maupun menyimpan hasil klasifikasi data dari media sosial Facebook, Twitter, dan Youtube.

4.2.1 Database Pemda

Untuk melakukan *streaming* data dari ketiga media sosial yaitu Facebook, Twitter dan Youtube, perlu adanya daftar akun yang akan dipanggil oleh *API*. Daftar akun tersebut disimpan dalam database berdasarkan setiap pemerintah daerah. Tabel 4.2.1 menunjukkan desain database yang digunakan untuk menyimpan daftar akun.

Tabel 4.2.1 Collection Pemda

Field	Tipe Data	Keterangan
_id	Integer	Nomor ID komentar
name	String	Nama Pemenrintah Daerah
facebook_resmi	String	Nama akun facebook resmi
facebook_influencer	String	Nama akun facebook influencer
twitter_resmi	String	Nama akun twitter resmi
twitter_influencer	String	Nama akun twitter influencer
youtube_resmi	String	Nama akun youtube resmi
youtube_influencer	String	Nama akun youtube influencer

4.2.2 Database Facebook

Data yang disimpan dari hasil klasifikasi Facebook pada penelitian ini adalah data komentar. Untuk dapat melakukan visualisasi data perlu menyimpan *field* yang diperlukan pada komentar Facebook.

Field dari komentar Facebook yang disimpan beserta tipe datanya dalam database MongoDB dapat dilihat pada Tabel 4.2.2.

Tabel 4.2.2 Collection Facebook *Comment*

Field	Tipe Data	Keterangan
_id	String	ID komentar facebook
page_id	String	Nama halaman akun
comment_createdDate	String	Waktu pengiriman komentar
comment_message	String	Isi pesan komentar

class	String	Hasil tipe klasifikasi
category	String	Hasil tipe kategori dinas

Pada penelitian ini yang akan divisualkan adalah jumlah setiap *class* masing-masing pemerintah daerah.

4.2.3 Database Twitter

Hasil klasifikasi Twitter pada penelitian ini adalah *tweet* balasan pada tweet yang dikirim oleh pemerintah dan *influencer*. Tabel 4.2.3 menunjukkan desain database untuk menyimpan hasil klasifikasi data dari Twitter.

Tabel 4.2.3 Collection Twitter *Comment*

Field	Tipe Data	Keterangan
_id	String	Nomor ID komentar
account_id	String	Nama akun
tweet_created Date	String	Waktu pengiriman komentar
text	String	Isi pesan komentar
class	String	Hasil tipe klasifikasi
category	String	Hasil tipe kategori dinas

Hasil visualisasi pada penelitian ini berdasarkan jumlah setiap *class* masing-masing pemerintah daerah.

4.2.4 Database Youtube

Data hasil klasifikasi dari komentar Youtube akan disimpan dalam database. Pada Tabel 4.2.4 menunjukkan desain database yang dibutuhkan untuk melakukan visualisasi hasil klasifikasi dari Youtube.

Tabel 4.2.4 Collection Youtube *Comment*

Field	Tipe Data	Keterangan
_id	String	ID komentar youtube

channel_id	String	ID channel
comment_createdDate	String	Waktu pengiriman komentar
comment_message	String	Isi pesan komentar
class	String	Hasil tipe klasifikasi
category	String	Hasil tipe kategori dinas

Jumlah setiap *class* masing-masing pemerintah daerah merupakan hasil visualisasi akhir pada penelitian ini.

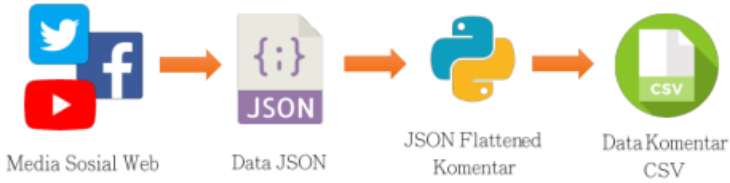
4.3 Pembuatan Model Klasifikasi

Pembuatan Model Klasifikasi bertujuan untuk membuat model yang nantinya akan digunakan untuk melakukan klasifikasi data ke dalam aplikasi. Ada 5 tahapan dalam pembuatan model klasifikasi yaitu *data crawling*, *data labeling*, *data pre-processing*, pembuatan model, dan *model validation.*

4.3.1 *Data Crawling*

Data crawling merupakan tahapan pengambilan data *post* dan komentar media sosial pemerintahan yang kemudian akan digunakan dalam proses klasifikasi. Tahap ini memanfaatkan id media sosial yang didaftar sebelumnya. Proses *crawling* ini menggunakan Python dengan memanfaatkan *API* yang disediakan oleh masing-masing media sosial untuk mendapatkan data JSON. API yang digunakan adalah *Graph API v2.12*, *Twitter API v1.1,* dan *Youtube API v3.* Data JSON tersebut dikonversikan menjadi file CSV untuk dilakukan proses pelabelan pada data. Jumlah data yang didapatkan sebesar 1084013 dari Facebook untuk periode Mei 2016 hingga Mei 2018, 136718 dari Twitter , dan 179665 dari youtube. Data yang digunakan sebanyak 6000 data dengan pemilihan secara

acak dari data ketiga media sosial. Gambaran proses crawling dijelaskan pada Gambar 4.3.1.



Gambar 4.3.1 Alur Proses *Crawling*

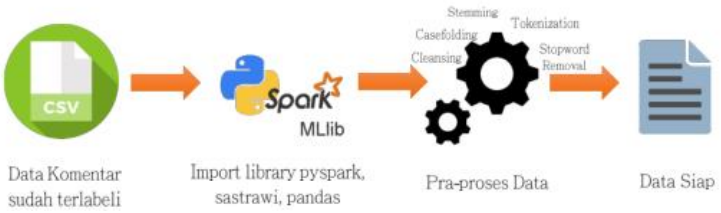
4.3.2 *Data Labeling*

Setelah proses *crawling* dilakukan, pemberian label setiap data diperlukan untuk pembuatan model klasifikasi. Data yang akan diberi label adalah data komentar netizen berbahasa indonesia. Pengecekan dilakukan dengan hanya mencocokkan kalimat dengan ada tidaknya kata bahasa Indonesia di dalamnya berdasarkan daftar kata bahasa Indonesia yang dibuat oleh Jim Geovedi [29]. Data tersebut nantinya akan diberikan label yang sesuai berdasarkan penelitan sebelumnya [5]. Jika pada hasil pelabelan terdapat hasil jumlah label yang sedikit, maka label akan dipertimbangkan untuk digabungkan dengan label lain atau dihilangkan. Proses *labeling* dilakukan oleh satu annotator yaitu penulis itu sendiri. Label bersifat *mutually exclusive label*. Jika terdapat label yang bersifat multilabel, maka penentuan label berdasarkan perspektif annotator. Gambaran tahap ini dijelaskan pada Gambar 4.3.2.

Gambar 4.3.2 Alur Proses Labeling

4.3.3 Data Pre-processing

Tahap *data pre-processing* mencakup beberapa langkah utama pengerjaan yakni pembersihan data dari tag atau karakter-karakter tertentu, pengubahan data menjadi huruf kecil *casefolding*, *stemming*, *stopword removal*, serta *tokenization*. Untuk penjelasan secara lebih detail. Proses ini menggunakan *library* dari Spark yaitu *MLlib* dan beberapa *library* lainnya. Gambaran tahap ini dijelaskan pada Gambar 4.3.3.



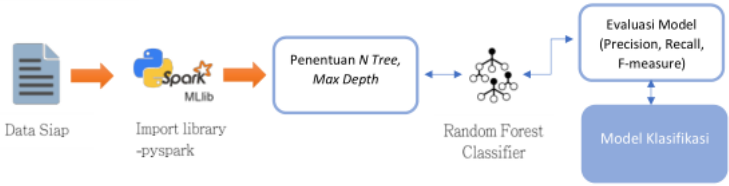
Gambar 4.3.3 Alur Pra-proses Data

1. *Cleansing* adalah tahap untuk menghilangkan karakter-karakter yang tidak terpakai pada data komentar, seperti angka, simbol, *hyperlink* dan menghapus beberapa row yang berisikan bukan bahasa Indonesia pada saat crawling.
2. *Casefolding* dilakukan untuk menyeragamkan struktur kata, pada penelitian ini menggunakan *lowercase* atau menjadi huruf kecil.
3. *Tokenization* dilakukan untuk memecah atau memotong string pada kalimat menjadi tiap kata yang menyusunnya.
4. *Stemming* dilakukan untuk menghilangkan kataimbuhan serta mengubah kata-kata pada data menjadi kata dasar yang memanfaatkan *library* Sastrawi.
5. *Stopwords Removal* dilakukan dengan mendaftar kata-kata dalam bahasa indonesia, merujuk pada penelitian

Fadillah Z Tala [30] yaitu dengan menghapus kata yang sering digunakan namun tidak memiliki nilai informasi.

4.3.4 Pembuatan Model

Pada tahapan ini, data siap proses akan dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Data siap tersebut dibagi dengan perbandingan 70% dan 30%. Data sebesar 70% akan digunakan sebagai data *training* dan 30% akan digunakan sebagai data *testing*. Data *training* digunakan untuk melatih pengetahuan terhadap model klasifikasi dan data testing digunakan untuk menguji kualitas model tersebut. Pembuatan model dilakukan perulangan dengan mengubah parameter sampai mendapatkan akurasi yang cukup baik. Perataan dataset juga dilakukan untuk mengetahui pengaruh banyak data tiap label yang digunakan. Gambaran dari tahap ini dijelaskan pada Gambar 4.3.4.



Gambar 4.3.4 Alur Pembuatan Model

4.3.5 Model Validation

Tahap validasi model memiliki tujuan untuk memastikan model yang dihasilkan dari hasil klasifikasi pada data *testing* sudah sesuai dengan label yang diberikan. Adapun beberapa hal yang dianalisis dalam tahap validasi model klasifikasi adalah

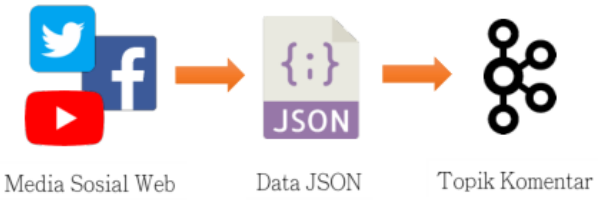
1. Hasil persentase *Precision* yang baik pada model klasifikasi.
2. Hasil persentase *Recall* yang baik pada model klasifikasi.
3. Hasil persentase *F-measure* yang baik pada model klasifikasi.

4.4 *Stream Classification and Visualization*

Stream Classification and Visualization pada penelitian ini merupakan tahap yang bertujuan untuk mencapai penelitian yang disesuaikan dengan komputasi secara otomatis untuk melakukakan klasifikasi beserta visualisasi data secara *real-time*. Pada tahap ini terbagi menjadi 5 tahapan yang akan dijelaskan sebagai berikut.

4.4.1 *Data Streaming*

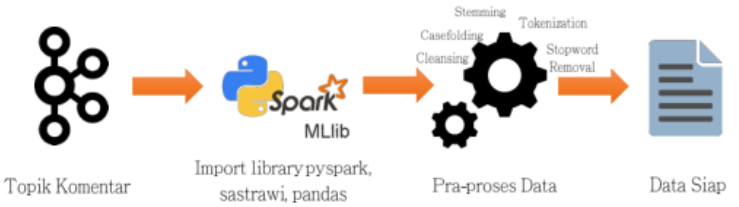
Data streaming merupakan tahapan pengambilan data *post* dan komentar pada akun sosial media resmi pemerintahan dan beserta *influencer* masing-masing daerah. Proses *streaming* pada penelitian ini menggunakan menggunakan *kafka* sebagai *data pipeline* dan *python* sebagai dasar bahasa pemrogramannya. *API* yang disediakan pada masing-masing media sosial digunakan untuk melakukan proses *streaming* untuk mendapatkan JSON yang berisikan data *post* dan komentar. Namun hanya *Twitter* yang menyediakan *Stream API*. Untuk melakukan *stream* data dari *Facebook* dan *Youtube* yang tidak menyediakan Stream API, maka proses looping pada Rest API dengan diberi waktu pengambilan menjadi satu-satunya opsi yang dijalankan pada penelitian ini. *Kafka* akan membagi data berdasarkan *topic*. Pada penelitian ini, *topic* yang akan di-*subscribe* adalah *topic* yang memilah data komentar. Proses pengambilan data dilakukan secara *real-time* ketika sistem aplikasi dijalankan. Gambaran proses *streaming* dijelaskan pada Gambar 4.4.1.



Gambar 4.4.1 Alur Proses Data Streaming

4.4.2 *Data Pre-processing*

Pada tahap ini, *topic* yang sebelumnya sudah di-*subscribe* akan membawa data komentar dan diteruskan melalui *spark*. Tahap *data pre-processing* pada tahap ini memiliki tahapan yang sama dengan proses *data pre-processing* saat pembuatan model dengan menggunakan *library* dari Spark yaitu *MLlib*, Sastrawi dan beberapa *library* lainnya. Gambaran tahap ini dijelaskan pada Gambar 4.4.2.



Gambar 4.4.2 Alur Praproses Data Streaming

4.4.3 Data Classification

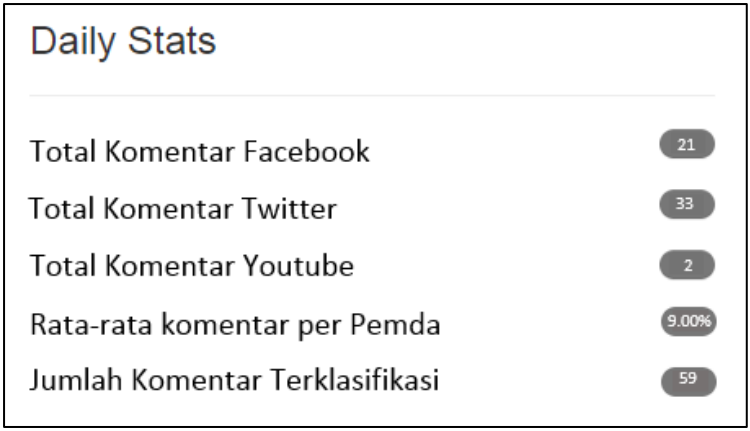
Data yang sudah siap akan dilaukan proses klasifikasi menggunakan algoritma *random forest* dengan model yang sudah dibuat sebelumnya. Proses ini menggunakan *library MlLib* dari *Pyspark*. Data yang dihasilkan akan disimpan dalam database untuk siap divisualkan. Gambaran proses klasifikasi pada data *stream* dijelaskan pada Gambar 4.4.3.

Gambar 4.4.3 Alur Klasifikasi Data *Stream*

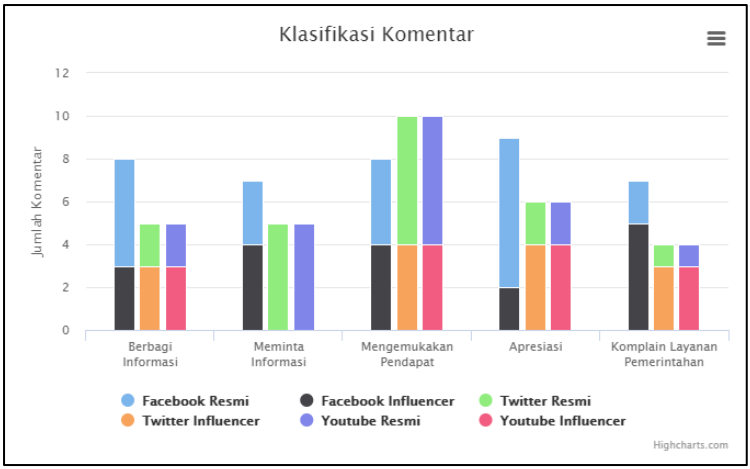
4.4.4 Desain Antarmuka Aplikasi Visualisasi

Melalui proses klasifikasi data didapatkan hasil klasifikasi yang siap divisualkan ke dalam beberapa kategori yang dihasilkan. Visualisasi yang digunakan dalam penelitian ini adalah

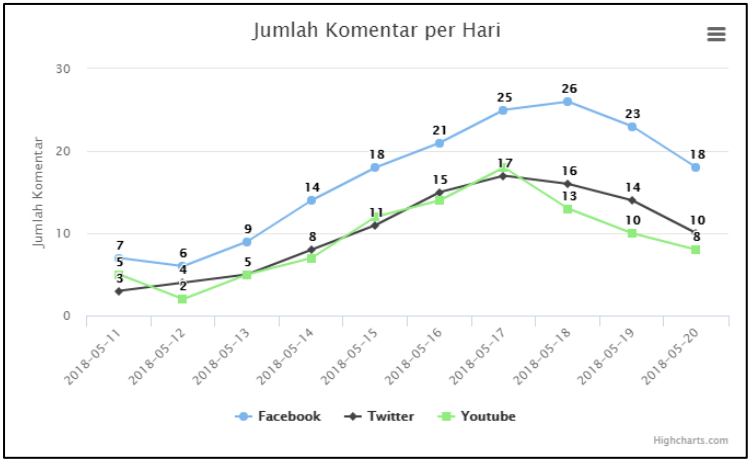
menggunakan bahasa pemrograman PHP, dimana user dapat melihatnya dalam sebuah dashboard. Rancangan dashboard yang akan ditampilkan dari analisa klasifikasi berdasarkan komentar tiap masing-masing pemerintah daerah yang dipilih. Pada Gambar 4.4.4 menjelaskan tentang *daily stats* jumlah data setiap media sosial yang berhasil terakuisisi, sedangkan Gambar 4.4.5 menampilkan pergerakan komentar per-hari yang berhasil diklasifikasian dan pada Gambar 4.4.6 rancangan tampilan grafik klasifikasi komentar, yang kemudian ditampilkan ke dalam aplikasi E-Govbench.



Gambar 4.4.4 Statistik Harian



Gambar 4.4.5 Visualisasi Klasifikasi Komentar



Gambar 4.4.6 Visualisasi Jumlah Komentar per Hari

BAB V
IMPLEMENTASI

Bab ini menjelaskan hasil dari implementasi perancangan studi kasus atau hasil dari proses pelaksanaan penelitian. Bagian implementasi akan menjelaskan mengenai lingkungan implementasi, pembuatan fitur-fitur aplikasi dalam bentuk kode, serta pengujian aplikasi.

5.1 Data Implementasi

Data yang digunakan dalam penelitian ini, baik itu data latih dan data uji, didapatkan dari hasil *crawling* media sosial milik 548 pemda yang ada di Indonesia. Dari data tersebut, sebesar 5907 diambil secara acak untuk dilabeli dan digunakan sebagai data latih dan uji untuk klasifikasi komentar netizen. Sebesar 2666 data diambil dari media sosial Facebook, 855 data dari Twitter, dan 2386 data dari Youtube yang dapat dilihat pada Tabel 5.1.1. Dari data mentah tersebut atribut yang diambil sudah dijelaskan pada bab 4 pada tiap-tiap media sosial. Kemudian masing-masing atribut tersebut ditambahkan atribut *category* yang merupakan atribut pelabelan kelas dari dataset yang ada. Hasil dari proses pelabelan, terdapat beberapa label mempunyai data yang tidak seimbang atau terpaut jauh jumlah data dengan label lainnya yang dapat dilihat pada Tabel 5.1.2. Ada 2 label yang dihilangkan yaitu Memasarkan Acara dan Pembentukan Komunitas atau Identitas dengan pertimbangan keterkaitan dengan pemerintahan yang tidak begitu kuat. Sedangkan ada label yang dapat digabungkan karena terdapat bias dalam pelabelan datanya yang banyak memiliki kesamaan yaitu Melaporkan Permalasahan Layanan, Komplain, Meminta Peingkatan Layanan, dan Meminta Layanan Baru dapat dilihat pada Gambar 5.1.1. Pada gambar tersebut terdapat komentar yang berisikan tentang melaporkan permasalahan namun juga memberikan komplain, melaporkan permasalahan layanan namun juga meminta peningkatan layanan dst. Tiap-tiap dataset dibagi menjadi data latih dan data uji dengan perbangingan

sebesar 70:30. Distribusi dataset yang digunakan dalam penelitian ini yang dijelaskan dalam Tabel 5.1.3.

Kapan aspal di pelintasan KA 78 & 81 dibetulin? Apa nunggu ada yg MATI dulu? Dasar ga ada otaknya lu semua @pemkotbekasi	Komplain
Yth. @BimaAryaS @PemkotaBogor mohon perhatian jalan berlubang di jembatan jalan sholeh iskandar (sebelum Delima Jaya) dapat segera diperbaiki karena membahayakan pengendara dan menghambat arus lalu lintas. Terima kasih.	Melaporkan Permasalahan Layanan
Ini bukan genangan biasa yang mampir setiap hujan di @Kota_Tangerang genangan di Perumahan Poris Indah @kel_cpi :-) Drainase tidak terhubung, pdhl got besar sudah ada, akhirnya air terus menggenang merepotkan warga apalagi untuk bayi dan manula. Semoga segera diatasi. @AboutTNG	Melaporkan Permasalahan Layanan
Bapak @pemkotdepok selamat yak udah ulang tahun 😂😂 semoga jd okeh kotanya , bikin alun" kek pak jangan apartemen mulu biar gak sumpek □□	Meminta Layanan Baru
Mohon diperbaiki pelayanan berkaitan dg nomor antrian @Dukcapil_Sragen Apabila terdpat krusakan pd mesin sgra di perbaiki oleh teknisi terkait, nomor antrian manual mjdkan sistem antri tdk obyektif. Terimakasih Cc : @sragenkab @MbakYuniBupati	Meminta Peningkatan Layanan
Antrian panjang di RSUD Tangsel. Bagaimana Bu @airinrachmi apakah bisa dibuat mengambil nomor antrian? Ini sangat tdk nyaman. Apalagi bnyk lansia yg berdiri,. Cc @humastangsel	Meminta Peningkatan Layanan
Dear @pemkotbekasi tolong dengan sangat jalanan bolong kayak gini gak pernah dibenerin! Udah lama banget bolong loh, tp kok gak ada perbaikan. Lokasi Jl. Terusan I Gusti Ngurah Rai (bawah kolong tol Cacing)	Komplain

Gambar 5.1.1 Data yang digabungkan

Tabel 5.1.1 Distribusi Dataset

Dataset	**Jumlah Data *Crawl***	**Jumlah Data yang diambil**
Dataset Facebook	1084013	2666
Dataset Twitter	136718	855
Dataset Youtube	179665	2386
Total	1400396	5907

Tabel 5.1.2 Distribusi Label

Label	**Jumlah Data**
Berbagi Informasi	1005
Meminta Informasi	1129
Mengemukakan Pendapat	1533
Melaporkan Permasalahan Layanan	172
Apresiasi	1111
Komplain	557
Meminta Peningkatan Layanan	195
Meminta Layanan Baru	81

Memasarkan Acara	66
Pembentukan Komunitas atau Identitas	55
Total	5907

Tabel 5.1.3 Distribusi Akhir Label

Label	Jumlah Data
Berbagi Informasi	1005
Meminta Informasi	1129
Mengemukakan Pendapat	1533
Apresiasi	1111
Komplain Layanan Pemerintahan	1008
Total	5787

5.2 Lingkungan Implementasi

Pada bagian ini dibahas terkait kriteria perangkat pengujian dalam implementasi penelitian. Adapun perangkat-perangkat yang dibutuhkan berupa perangkat keras dan spesifikasinya ditunjukkan dengan Tabel 5.2.1. Kemudian untuk perangkat lunak yang digunakan dalam implementasi model ditunjukkan dalam Tabel 5.2.2. Selain menggunakan hardware dan software, dalam penelitian ini juga digunakan beberapa library untuk mendukung proses klasifikasi menggunakan python, adapun beberapa library yang digunakan ditunjukkan dalam Tabel 5.2.3.

Tabel 5.2.1 Spesifikasi Perangkat Keras

No.	Hardware	Spesifikasi
1	Jenis	Lenovo Z410
2	Processor	Intel Core i7-4702MQ
3	RAM	DDR4 8GB
4	Harddisk	1TB HDD 5400/7200

Tabel 5.2.2 Perangkat Lunak yang digunakan

No.	Software	Penggunaan
1	Ubuntu 16.04	Sistem Operasi

2	Apache 2.4.33 & PHP 7.2.5	Webserver
3	Python 3.6 32-bit	Pemrosesan Data
4	Jupyter Notebook	Python IDE
5	MongoDB 3.6	Database Server
6	Google Chrome 67.0.3396.79	Web Browser
7	Sublime Text 3 & Atom 1.27.2	Editor

Tabel 5.2.3 Library Python yang digunakan

No.	Library	Penggunaan
1	pandas 0.23.0	Dataframe
2	Sastrawi 1.1.0	Stemming
3	pyspark 2.2.1	Pra-proses & Klasifikasi (Random Forest)
4	re & string	Pra-proses
5	json	JSON Flattened

Tabel 5.2.4 Konfigurasi Spark Streaming

No.	Konfigurasi	Nilai	Deskripsi
1	streamingcontext	45s	Interval setiap *batch*
2	spark.streaming.backpressure.enabled	True	Mengontrol *receiving rate* berdasarkan *current batch scheduling delays* dan waktu proses.
3	spark.streaming.backpressure.initialRate	50	Maksimum *receiving rate* setiap *receiver* untuk *batch* pertama
4	spark.streaming.kafka.maxRatePerPartition	50	Maksimum rate (jumlah record per detik) data dari setiap partisi Kafka

Tabel 5.2.4 menjelaskan konfigurasi pada Spark Streaming yang digunakan. Proses *streaming* pada Spark menggunakan

sistem *micro-batching* dengan durasi sesuai interval pada konfigurasi. Proses *micro-batching* berguna untuk menstabilkan kinerja CPU karena data *streaming* dari Facebook dan Youtube cukup cepat. Jika data tersebut diproses pada saat itu juga akan memperberat CPU sehingga memperlambat server. Penentuan *rate* data dari Kafka juga diperlukan agar data tidak overload pada saat di *batch.* Adapun 3 topik yang di-*subscribe* pada Kafka sehingga proses *micro-batching* sangat diperlukan pada aplikasi *streaming* ini. Untuk konfigurasi Kafka dijelaskan pada tugas akhir yang dikerjakan oleh Guntur Kondang Prakoso [31]. Selain itu untuk konfigurasi *database* MongoDB dijelaskan pada tugas akhir yang dikerjakan oleh Dyaksa Hanindito [32] dan konfigurasi laravel dijelaskan pada tugas akhir yang dikerjakan oleh Nody Risky Pratomo [33].

5.3 *Data Crawler*

5.3.1 Pengambilan Data Akun

Untuk melakukan *crawling* dari ketiga media sosial, diperlukan akun dari setiap pemerintahan untuk mendapatkan data yang diinginkan. Daftar akun pada setiap pemda untuk proses *crawling* didapatkan dari google spreadshet yang sudah didaftarkan akun-akun setiap pemda yang dapat dilihat pada Gambar 5.3.1. Adapun Kode 5.3.1 untuk mendapatkan data pada kolom yang diinginkan di *google spreadsheet.*

Provinsi/Kabupaten	Facebook	Twitter	Youtube	Facebook Influencer	Twitter Influencer
PROVINSI NAD	326019957540137	acehprov		acehinfo	TweetAneukAceh
KAB. ACEH SELATAN				116815498393415	
KAB. ACEH TENGGARA					
KAB. ACEH TIMUR					AcehTimurNAD
KAB. ACEH TENGAH	DinasKominfoAcehTengah	diskominfoat			
KAB. ACEH BARAT					Kabar_AB
KAB. ACEH BESAR					
KAB. PIDIE	pemerintahkabupatenpidie				

Gambar 5.3.1 Daftar Akun Pemda

```
1.  import gspread
2.  from oauth2client.service_account import Service
    AccountCredentials
3.
```

```
scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']

creds = ServiceAccountCredentials.from_json_keyfile_name('client_secret.json', scope)
client = gspread.authorize(creds)

def scrapingcolumn(col_number, fromrow=0, torow=None):

    sheet = client.open('situs.csv').sheet1
    result = sheet.col_values(col_number)

    return result[fromrow:torow]
```

Kode 5.3.1 Mengambil Akun dari *Google Spreadsheets*

5.3.2 Facebook *Crawler*

Dalam membuat Facebook *Crawler* diperlukan kustomisasi dari implementasi *Graph API* untuk mendapatkan komentar. Data komentar akan didapatkan setelah data posting pada halaman facebook sudah didapatkan. *Crawler* yang akan dibuat akan melakukan pengambilan data 313 akun Facebook mulai dari Mei 2016 hingga Mei 2018.

Untuk membuat Facebook *Crawler* ini terdiri dari 1 file python yang mana mempunyai anggota 1 *main method*, 7 *method* lainnya. Namun hanya 5 *method* dalam *crawler* ini yang berkaitan dengan penelitian. Berikut adalah penjelasan dari masing-masing *method* yang berkaitan.

```
def scrapeFacebookPageFeedStatus(page_id, access_token, since_date, until_date, number, pemdaname, type):
        has_next_page = True
        num_processed = 0
        scrape_starttime = datetime.datetime.now()
        after = ''
        base = "https://graph.facebook.com/v2.12"
        node = "/{}".format(page_id)
```

```
8.          parameters = "?access_token={}&fields=id
    ,name,fan_count,posts.limit({})".format(access_t
    oken, 100)
9.          since = ".since({})".format(since_date)
    if since_date \
10.             is not '' else ''
11.         until = ".until({})".format(until_date)
    if until_date \
12.             is not '' else ''
13.
14.         print("Scraping No.{} :\n Pemda : {}\n T
    ype : {}\n Page ID : {}\n at : {}\n".format(numb
    er, pemdaname, type, page_id, scrape_starttime))

15.
16.         complete_dict = {}
17.         complete_dict['page_id_name'] = page_id

18.         complete_dict['post'] = []
19.
20.         base_url = base + node + parameters + si
    nce + until
21.
22.         url = getFacebookPageFeedUrl(base_url)
23.
24.         j_input = json.loads(request_until_succe
    ed(url).decode())
25.
26.         complete_dict['page_id_number'] = j_inpu
    t['id']
27.         complete_dict['page_name'] = j_input['na
    me']
28.         complete_dict['page_fanCount'] = j_input
    ['fan_count']
29.           while has_next_page:
30.
31.             after = '' if after is '' else ".aft
    er({})".format(after)
32.             base_url = base + node + parameters
    + since + until + after
33.
34.             url = getFacebookPageFeedUrl(base_ur
    l)
35.
```

```
            j_input = json.loads(request_until_s
ucceed(url).decode())
            if 'posts' in j_input:
                after = j_input['posts']['paging
']['cursors']['after']
                comments = getCommentsForStatuse
s(base_url)
                for status in j_input['posts']['
data']:
                    status_data = processFaceboo
kPageFeedStatus(status)
                    status_data['post_comments']
 = comments[status_data['post_id']]
                    complete_dict['post'].append
(status_data)
                    num_processed += 1
                    if num_processed % 100 == 0:

                        print("{} Statuses Proce
ssed: {}".format
                              (num_processed, da
tetime.datetime.now()))

            else:
                has_next_page = False
        return complete_dict
```

Kode 5.3.2 Pengambilan Postingan Facebook

Kode 5.3.2 menunjukkan isi kode dari *method* scrapeFacebookPageFeedStatus yang berfungsi untuk mengambil data post di halaman Facebook, yang dilakukan sebanyak nilai interval yang didapat dari selisih waktu mulai dengan waktu akhir pengambilan data. Waktu mulai dan waktu akhir pengambilan data didefinisikan di *main method*. Untuk membaca *post* dari sebuah akun maka dibutuhkan *request* URL berdasarkan URL serta *field* yang dibutuhkan antara lain *id, name, fan_count,* dan *posts*. Agar dapat mengambil dan memodikfikasi data yang diinginkan, URL yang didapatkan dapat dimuat dalam bentuk JSON untuk dibuat sebuah *dictionary* sendiri. Pada *method* ini juga berisi pemanggilan method getCommentsForStatuses yang digunakan untuk menagmbil komentar. Adapun data yang tidak didapatkan dari

request URL seperti id pemda dan nama pemda yang dibuthkan untuk memberikan identitas kepemilikan data yang diambil dari halaman Facebook di penelitian ini.

```
1.  def getCommentsForStatuses(base_url):
2.      comments_dict = {}
3.
4.      fields = "{id,comments}"
5.      url = base_url + fields
6.      data = json.loads(request_until_succeed(url)
    .decode())['posts']['data']
7.
8.      for status in data:
9.          id = status['id']
10.         komentar = []
11.
12.         if 'comments' in status:
13.             komentar = status['comments']['data'
    ]
14.             paging = status['comments']['paging'
    ]
15.
16.             if 'next' in paging:
17.                 nextcommentpage = status['commen
    ts']['paging']['next']
18.                 comments_has_next_page = True
19.
20.                 while comments_has_next_page:
21.                     nextcommentpage = '' if next
    commentpage is '' else "{}".format(nextcommentpa
    ge)
22.
23.                     crawlnextpage = json.loads(r
    equest_until_succeed(nextcommentpage).decode())
24.                     komentar = komentar + crawln
    extpage['data']
25.
26.                     if crawlnextpage['data']:
27.                         if 'next' in crawlnextpa
    ge['paging']:
28.                             nextcommentpage = cr
    awlnextpage['paging']['next']
29.                         else:
```

```
                                comments_has_next_pa
    ge = False
                        else:
                            comments_has_next_page =
     False

            comments_dict[id] = komentar

        return comments_dict
```

Kode 5.3.3 Pengambilan Komentar Facebook

Kode 5.3.3 menunjukkan isi kode dari *method* getCommentsForStatuses yang berfungsi untuk mengambil data komentar pada setiap post di halaman *Facebook*, yang dipanggil didalam *method* pengambilan post sehingga dapat mengetahui id post yang akan diambil komentarnya. Untuk membaca komentar dari sebuah post juga dibutuhkan *request* URL berdasarkan URL serta *field* yang dibutuhkan antara lain *id*, dan *comments*. Data ini juga dimasukkan kedalam *dictionary* agar mengetahui kepimilikan komentar dalam data JSON.

```
if __name__ == '__main__':
    start_row = 1

    namapemda = spreadsheet.scrapingcolumn(1, st
art_row, None)
    idpemda = spreadsheet.scrapingcolumn(3, star
t_row, None)
    idpagepemda = spreadsheet.scrapingcolumn(7,
start_row, -1)
    idpageinfluencer = spreadsheet.scrapingcolum
n(10, start_row, -1)

    since_date = "2018-05-01"
    until_date = ""

    access_token = <Access Token Facebook>

    for satuanidpemda in idpemda[298:]:

        pemda_dict = {}
        pemda_dict['pemda_id'] = satuanidpemda
```

```
18.         pemda_dict['pemda_name'] = namapemda[idp
    emda.index(satuanidpemda)]
19.         pemda_dict['resmi'] = {}
20.         pemda_dict['influencer'] = {}
21.
22.         satuanidpagepemda = idpagepemda[idpemda.
    index(satuanidpemda)]
23.         satuanidinfluencer = idpageinfluencer[id
    pemda.index(satuanidpemda)]
24.
25.         if satuanidpagepemda is not '':
26.             pemda_dict['resmi'] = scrapeFacebook
    PageFeedStatus(satuanidpagepemda, access_token,
    since_date, until_date, satuanidpemda, pemda_dic
    t['pemda_name'], 'resmi')
27.
28.         elif satuanidpagepemda is '':
29.            pemda_dict['resmi']['post'] = []
30.            pemda_dict['resmi']['post'].append({'
    post_comments': []})
31.
32.         if satuanidinfluencer is not '':
33.             pemda_dict['influencer'] = scrapeFac
    ebookPageFeedStatus(satuanidinfluencer, access_t
    oken, since_date, until_date, satuanidpemda, pem
    da_dict['pemda_name'], 'influencer')
34.
35.         elif satuanidinfluencer is '':
36.             pemda_dict['influencer']['post'] = [
    ]
37.             pemda_dict['influencer']['post'].app
    end({'post_comments': []})
38.
39.         if satuanidpemda is idpemda[0]:
40.             exported = '[' + str(json.dumps(pemd
    a_dict, indent=4)) + ','
41.
42.             with open('facebook_lengkap.json', '
    a') as file:
43.                 file.write(exported)
44.
45.         elif satuanidpemda is not idpemda[-1]:
46.             exported = str(json.dumps(pemda_dict
    , indent=4)) + ','
47.
```

```
48.             with open('facebook_lengkap.json', '
    a') as file:
49.                 file.write(exported)
50.
51.         else:
52.             exported = str(json.dumps(pemda_dict
    , indent=4)) + ']'
53.
54.             with open('facebook_lengkap.json', '
    a') as file:
55.                 file.write(exported)
```

Kode 5.3.4 Main Method Facebook Crawler

Kode 5.3.4 merupakan *main method* dari keseluruhan kode untuk Facebook *Crawler*. Akun keseluruhan akan dimuat dalam bentuk *list* diamana data akun Facebook didapatkan dari penarikan data melalui *google spreadsheets* yang sebelumnya dibuat. Inisiasi waktu mulai dan waktu akhir *crawl* juga dilakukan di *main method*. *Crawler* akan berjalan satu-satu dengan perulangan sesuai *list* semua akun yang sebelumnya dibuat. Disini juga melakukan pengecekan terhadap pemerintah daerah yang memliki aku resmi namun tidak memiliki akun influencer dan sebaliknya. Data akhir dalam proses *crawling* ini disimpan dengan nama file facebook_lengkap.json yang berisikan JSON lengkap keseluruhan akun pemerintah daerah. Contoh JSON dapat dilihat pada Tabel 4.1.2.

5.3.3 Twitter *Crawler*

Pembuatan *crawler* untuk Twitter sedikit berbeda dengan Facebook. Jika pada *Facebook* harus mengambil induk *post* untuk mengambil komentar, *crawler* pada *Twitter* tidak perlu untuk mengambil induk *tweet* dimana *tweet* tersebut dibalas atau dikomentar.

Pada pengambilan data Tweet tidak diberi waktu awal dan akhir *crawling* seperti pada Facebook, dikarenakan data dari Twitter jauh lebih sedikit daripada Facebook. Adapun *method* dan *main method* pada crawler ini akan dijelaskan sebagai berikut.

```
1.  def crawltweets(akun, tipe, complete_dict):
2.
3.      num_processed = 0
4.      complete_dict[tipe] = {}
5.
6.      complete_dict[tipe]['account_name'] = akun
7.      complete_dict[tipe]['retweet_count'] = 0
8.      complete_dict[tipe]['favorite_count'] = 0
9.      complete_dict[tipe]['tweets'] = []
10.     complete_dict[tipe]['tweets_reply'] = []
11.     try:
12.         for tweets in tweepy.Cursor(
13.                 twitterAPI.search,
14.                 q='@' + akun,
15.                 include_rts=True,
16.                 tweet_mode='extended').items():
17.
18.             json_str = json.dumps(tweets._json)
19.             j_results = json.loads(json_str)
20.
21.             ambiljson(j_results, complete_dict,
    akun, tipe)
22.
23.             num_processed += 1
24.             if num_processed % 100 == 0:
25.                 print("{} Tweet Processed: {}".f
    ormat(num_processed,
26.
          datetime.datetime.now()))
27.         crawlaccounts(akun, num_processed, compl
    ete_dict, tipe)
28.         complete_dict[tipe]['reply_count'] = num
    _processed
29.     except tweepy.TweepError as e:
30.         logging.exception("message")
```

Kode 5.3.5 Pengambilan Data Tweet Reply

Kode 5.3.5 menunjukkan isi kode dari *method* crawltweets dimana untuk mengambil tweet yang merupakan balasan dari suatu tweet induk. *API* yang digunakan adalah *API Search* dimana dengan peraturan *extended tweet mode*. Pengaturan ini

berfungsi untuk mengambil seluruh teks tweet yang jika tweet mode ini tidak diinisiasi maka teks dapat terpotong.

```
def ambiljson(j_results, complete_dict, akun, tipe):

    if 'RT @' not in j_results['full_text']:
        complete_tweet = {}

        complete_tweet['tweet_id'] = j_results['id_str']
        complete_tweet['tweet_message'] = j_results['full_text']
        complete_tweet['created_at'] = j_results['created_at']

        if j_results['user']['screen_name'] == akun:
            complete_dict[tipe][
                'retweet_count'] = complete_dict[tipe]['retweet_count'] + j_results['retweet_count']

        if j_results['user']['screen_name'] == akun:
            complete_dict[tipe][
                'favorite_count'] = complete_dict[tipe]['favorite_count'] + j_results['favorite_count']

        if 'media' in j_results['entities']:
            complete_tweet['tweet_type'] = j_results['entities']['media'][0][
                'type']
        else:
            complete_tweet['tweet_type'] = "text"

        complete_dict['reply_count'] += 1

        if j_results['user']['screen_name'] == akun:
            complete_dict[tipe]['tweets'].append(complete_tweet)
```

```
28.         else:
29.             complete_dict[tipe]['tweets_reply'].
    append(complete_tweet)
30.
31.     return complete_dict
```

Kode 5.3.6 Membuat JSON Dictionary

Kode 5.3.6 menunjukkan isi kode dari *method* ambiljson dimana pada crawler ini sama seperti Facebook, data dari twitter akan dipilah dimasukkan kedalam dictionary terlebih dahulu untuk hanya mengambil data yang diperlukan saja.

```
1.  if __name__ == '__main__':
2.
3.      start_row = 1
4.
5.      id_pemda_list = spreadsheet.scrapingcolumn(3
    , start_row, None)
6.      nama_pemda_list = spreadsheet.scrapingcolumn
    (4, start_row, None)
7.      account_list = spreadsheet.scrapingcolumn(8,
     start_row, -1)
8.      account_influencer_list = spreadsheet.scrapi
    ngcolumn(11, start_row, -1)
9.
10.     access_token = <Twitter Access Token>
11.     access_token_secret = <Twitter Access Token
    Secret>
12.     consumer_key = <Twitter Consumer Key>
13.     consumer_secret = <Twitter Consumer Secret>
14.
15.     authHandler = tweepy.OAuthHandler(consumer_k
    ey, consumer_secret)
16.     authHandler.set_access_token(access_token, a
    ccess_token_secret)
17.     twitterAPI = tweepy.API(
18.         authHandler, wait_on_rate_limit=True, wa
    it_on_rate_limit_notify=True)
19.
20.     for id_pemda in id_pemda_list:
21.
22.         complete_dict = {}
23.
```

```
24.         complete_dict['pemda_id'] = id_pemda
25.
26.         nama_pemda = nama_pemda_list[id_pemda_li
    st.index(id_pemda)]
27.         complete_dict['pemda_name'] = nama_pemda

28.
29.         account = account_list[id_pemda_list.ind
    ex(id_pemda)]
30.         account_influencer = account_influencer_
    list[id_pemda_list.index(id_pemda)]
31.
32.         if account is not '':
33.
34.             account_id = account_id_list[id_pemd
    a_list.index(id_pemda)]
35.             complete_dict['account_id'] = accoun
    t_id
36.
37.             crawltweets(account, complete_dict,
    'resmi')
38.             if account_influencer is not '':
39.                     crawltweets(accounts_influen
    cer, complete_dict, 'influencer')
40.                 else:
41.                     tipe = "influencer"
42.                     nullValues(complete_dict, ti
    pe)
43.
44.         else:
45.             tipe = "resmi"
46.             nullValues(complete_dict, tipe)
47.             crawltweets(accounts_influencercompl
    ete_dict, 'influencer')
48.
49.         exported = str(json.dumps(complete_dict,
     indent=4))
50.
51.         if id_pemda == id_pemda_list[0]:
52.
53.             with open('twitter_lengkap_mei.json'
    , 'w') as file:
54.                 file.write('[' + exported + ',')

55.
```

```
56.         elif id_pemda == id_pemda_list[-1]:
57.
58.             with open('twitter_lengkap.json', 'a
    ') as file:
59.                 file.write(exported + ']')
60.
61.         else:
62.             with open('twitter_lengkap.json', 'a
    ') as file:
63.                 file.write(exported + ',')
```

Kode 5.3.7 Main Method Twitter Crawler

Kode 5.3.7 merupakan *main method* dari keseluruhan kode untuk Twitter *Crawler*. Akun keseluruhan akan dimuat dalam bentuk *list* diamana data akun *twitter* didapatkan dari penarikan data melalui *google spreadsheets* yang sebelumnya dibuat. Dalam method ini juga menghubungkan *client* dengan *twitter API* menggunakan *OAuthHandler* yang dibantu dengan menggunakan *library tweepy*. Sama seperti Facebook, *crawler* akan berjalan satu-satu dengan perulangan sesuai *list* semua akun yang sebelumnya dibuat. Disini juga melakukan pengecekan terhadap pemerintah daerah yang memliki akun resmi namun tidak memiliki akun influencer dan sebaliknya. Data akhir dalam proses *crawling* ini disimpan dengan nama file twitter_lengkap.json yang berisikan JSON lengkap keseluruhan akun pemerintah daerah. Contoh JSON dapat dilihat padaTabel 4.1.3.

5.3.4 *Youtube Crawler*

Pembuatan *crawler* untuk Youtube tidak jauh berbeda dengan dengan Facebook. Kesamaannya pada Facebook harus mengambil induk *post* untuk mengambil komentar, dalam *crawler* Youtube juga perlu untuk mengambil induk *video* dimana komentar akan diambil.

Untuk perbedaannya adalah dari akun yang terdaftar pada *google spreadsheet*. Ada 2 tipe akun yaitu berupa id channel dan username. Pada pengambilan data Youtube, sama seperti Twitter tidak diberi waktu awal dan akhir *crawling.* Adapun

method dan *main method* pada crawler ini akan dijelaskan sebagai berikut.

```
1.  def channelusernametoid(idchannel):
2.        parameters = {"part": "snippet",
3.                      "forUsername": idchannel,
4.                      "key": api_key,
5.                      "fields": "items(id,snippet/
    title)"}
6.        url = "https://www.googleapis.com/youtube/
    v3/channels"
7.        usernameconverter = {}
8.
9.        page = requests.request(method='get', url=
    url, params=parameters)
10.       j_results = json.loads(page.text)
11.       usernameconverter['title'] = j_results['it
    ems'][0]['snippet']['title']
12.       usernameconverter['id'] = j_results['items
    '][0]['id']
13.
14.       return usernameconverter
```

Kode 5.3.8 Mengubah Username ke Channel Id

Kode 5.3.8 menunjukkan isi kode dari *method* channelusernametoid dimana untuk mengubah *username* menjadi id channel. Hal ini berguna untuk melakukan *crawling* karena *API* yang disediakan oleh Youtube yaitu melakukan *crawling* untuk mendapatkan *list* video pada satu akun harus menggunakan channel id.

```
1.  def channelnameforfilename(idchannel):
2.        parameters = {"part": "snippet",
3.                      "id": idchannel,
4.                      "key": api_key,
5.                      "fields": "items/snippet/tit
    le"}
6.        url = "https://www.googleapis.com/youtube/
    v3/channels"
7.        page = requests.request(method='get', url=
    url, params=parameters)
8.        j_results = json.loads(page.text)
```

```
9.
10.       idconverter = {}
11.
12.       if j_results['items']:
13.             idconverter['title'] = j_results['it
    ems'][0]['snippet']['title']
14.             idconverter['id'] = idchannel
15.       else:
16.             idconverter = channelusernametoid(id
    channel)
17.       return idconverter
```

Kode 5.3.9 Mengambil Nama Channel

Kode 5.3.9 menunjukkan isi kode dari *method* channelnameforfilename dimana untuk membuat identitas dari satu akun. Dalam *method* inilah dilakukan pengecekan apakah akun tersebut masih berupa username atau sudah berupa *channel id*. Tujuan lainnya adaah mendapatkan nama channel agar nantinya lebih mudah mengenali kepemilikan komentar.

```
1.  def crawlcomments(idvideo):
2.        comm_dict_returned = []
3.        parameters = {"part": "snippet",
4.                      "maxResults": 100,
5.                      "videoId": idvideo,
6.                      "key": api_key,
7.                      "fields": "items(snippet(top
    LevelComment(id,snippet(publishedAt,textOriginal
    ))),nextPageToken"
8.                      }
9.        url = "https://www.googleapis.com/youtube/
    v3/commentThreads"
10.       nextPageToken = ''
11.       has_next_page = True
12.
13.       while has_next_page:
14.             parameters['pageToken'] = '' if '' e
    lse nextPageToken
15.
16.             try:
17.               page = requests.request(method="ge
    t", url=url, params=parameters)
18.               page.raise_for_status()
```

```
19.
20.                 comment_results = json.loads(page.
    text)
21.
22.                 if comment_results['items']:
23.
24.                   for comment in comment_results['
    items']:
25.                         comm_dict = {}
26.                         comm_dict['comment_id'] =
    comment['snippet']['topLevelComment']['id']
27.                         comm_dict['comment_message
    '] = comment['snippet']['topLevelComment']['snip
    pet']['textOriginal']
28.                         comm_dict['comment_date']
    = comment['snippet']['topLevelComment']['snippet
    ']['publishedAt']
29.                         comm_dict_returned.append(
    comm_dict)
30.
31.                 if 'nextPageToken' in comment_resu
    lts:
32.                       nextPageToken = comment_resu
    lts['nextPageToken']
33.                 else:
34.                       has_next_page = False
35.
36.               except requests.HTTPError as e:
37.                 has_next_page = False
38.                 pass
39.
40.       return comm_dict_returned
```

Kode 5.3.10 Pengambilan Komentar Youtube

Kode 5.3.10 menunjukkan isi kode dari *method* crawlcomments yang berfungsi untuk mengambil data komentar pada setiap video di channel Youtube, yang dipanggil didalam *method* pengambilan video sehingga dapat mengetahui id video yang akan diambil komentarnya. Untuk membaca komentar dari sebuah post juga dibutuhkan *request* URL berdasarkan URL serta *field* yang dibutuhkan yaitu *id*, dan *textOriginal*. Data ini juga dimasukkan kedalam *dictionary* agar mengetahui kepimilikan komentar dalam data JSON.

```
1.  def crawlvideo(api_key, channelId, pemdaID, pemd
    aName, channelType):
2.    temp = channelnameforfilename(channelId)
3.
4.    channelName = temp['title']
5.    idchannel = temp['id']
6.
7.    parameters = {"part": "snippet",
8.                  "channelId": idchannel,
9.                  "maxResults": 50,
10.                 "publishedAfter": "2018-05-
    01T00:00:00Z",
11.                 "key": api_key,
12.                 "type": "video",
13.                 "fields": "items(id/videoId,snip
    pet/channelTitle),nextPageToken"}
14.   url = "https://www.googleapis.com/youtube/v3/s
    earch"
15.   video_search_nextpage = True
16.   num_processed = 0
17.   scrape_starttime = datetime.datetime.now()
18.
19.   complete_dict = {}
20.   complete_dict['channel_id'] = parameters['chan
    nelId']
21.   complete_dict['video'] = []
22.
23.   while video_search_nextpage:
24.         page = requests.request(method='get', ur
    l=url, params=parameters)
25.         j_results = json.loads(page.text)
26.
27.         if j_results['items']:
28.
29.               for video in j_results['items']:
30.
31.                     complete_dict['channel_name'
    ] = video['snippet']['channelTitle']
32.
33.                     complete_video = {}
34.
35.                     complete_video['video_id'] =
     video['id']['videoId']
36.
37.                     returned_statistics = {}
```

```
38.                     returned_statistics = crawls
    tatistics(video['id']['videoId'])
39.
40.                     complete_video['video_title'
    ] = returned_statistics[complete_video['video_id
    ']]['video_title']
41.                     complete_video['video_date']
     = returned_statistics[complete_video['video_id'
    ]]['video_date']
42.
43.                     statistics = returned_statis
    tics[complete_video['video_id']]['video_statisti
    c']
44.                     complete_video['video_dislik
    eCount'] = int(statistics['dislikeCount']) if 'd
    islikeCount' in statistics else 0
45.                     complete_video['video_likeCo
    unt'] = int(statistics['likeCount']) if 'likeCou
    nt' in statistics else 0
46.                     complete_video['video_viewCo
    unt'] = int(statistics['viewCount']) if 'viewCou
    nt' in statistics else 0
47.                     complete_video['video_commen
    tCount'] = int(statistics['commentCount']) if 'c
    ommentCount' in statistics else 0
48.
49.                     returned_comments = {}
50.                     returned_comments = crawlcom
    ments(video['id']['videoId'])
51.                     complete_video['video_commen
    ts'] = returned_comments
52.
53.                     complete_dict['video'].appen
    d(complete_video)
54.
55.                     num_processed += 1
56.                     if num_processed % 10 == 0:

57.                         print("{} Video Processe
    d: {}".format
58.                               (num_processed, da
    tetime.datetime.now()))
59.
60.         if 'nextPageToken' in j_results:
```

```
61.                 parameters['pageToken'] = j_result
    s['nextPageToken']
62.         else:
63.                 video_search_nextpage = False
64.
65.   return complete_dict
```

Kode 5.3.11 Pengambilan Informasi Video Youtube

Kode 5.3.11 menunjukkan isi kode dari *method* crawlvideo yang berfungsi untuk mengambil data video di channel Youtube Untuk mengambil data video dari sebuah channel maka dibutuhkan *request* URL berdasarkan URL serta *field* yang diinginkan. Sama seperti *crawler* lainny, untuk mengambil dan memodikfikasi data yang diinginkan, URL yang didapatkan dapat dimuat dalam bentuk JSON untuk dibuat sebuah *dictionary* sendiri. Pada *method* ini juga berisi pemanggilan method crawlcomments yang digunakan untuk menagmbil komentar. Adapun data yang tidak didapatkan dari *request* URL seperti channel id dan nama pemda yang dibutuhkan untuk memberikan identitas kepemilikan data yang diambil dari channel *Youtube* di penelitian ini.

```
1.  if __name__ == '__main__':
2.
3.    start_row = 1
4.
5.    idpemda = spreadsheet.scrapingcolumn(3, start_
    row, None)
6.    namapemda = spreadsheet.scrapingcolumn(1, star
    t_row, None)
7.    idchannelpemda = spreadsheet.scrapingcolumn(9,
     start_row, -1)
8.    idchannelinfluencer = spreadsheet.scrapingcolu
    mn(12, start_row, -1)
9.
10.   api_key = <Youtube API Key>
11.
12.   for satuanidpemda in idpemda:
13.
14.     pemda_dict = {}
15.     pemda_dict['pemda_id'] = satuanidpemda
```

```
16.     pemda_dict['pemda_name'] = namapemda[idpemda
    .index(satuanidpemda)]
17.     pemda_dict['resmi'] = {}
18.     pemda_dict['influencer'] = {}
19.
20.     satuanidchannelpemda = idchannelpemda[idpemd
    a.index(satuanidpemda)]
21.     satuanidchannelinfluencer = idchannelinfluen
    cer[idpemda.index(satuanidpemda)]
22.     if satuanidchannelpemda is not '':
23.       pemda_dict['resmi'] = crawlvideo(api_key,
    satuanidchannelpemda, pemda_dict['pemda_id'], pe
    mda_dict['pemda_name'], 'resmi')
24.
25.     elif satuanidchannelpemda is '':
26.       pemda_dict['resmi']['video'] = []
27.       pemda_dict['resmi']['video'].append({'vide
    o_comments': []})
28.
29.     if satuanidchannelinfluencer is not '':
30.       pemda_dict['influencer'] = crawlvideo(api_
    key, satuanidchannelinfluencer, pemda_dict['pemd
    a_id'], pemda_dict['pemda_name'], 'influencer')

31.
32.     elif satuanidchannelinfluencer is '':
33.       pemda_dict['influencer']['video'] = []
34.       pemda_dict['influencer']['video'].append({
    'video_comments': []})
35.
36.     if satuanidpemda is idpemda[0]:
37.       exported = '[' + str(json.dumps(pemda_dict
    , indent=4)) + ','
38.
39.       with open('youtube_lengkap.json', 'a') as
    file:
40.           file.write(exported)
41.
42.     elif satuanidpemda is not idpemda[-1]:
43.       exported = str(json.dumps(pemda_dict, inde
    nt=4)) + ','
44.
45.       with open('youtube_lengkap.json', 'a') as
    file:
46.           file.write(exported)
```

```
47.
48.     else:
49.       exported = str(json.dumps(pemda_dict, inde
    nt=4)) + ']'
50.
51.       with open('youtube_lengkap_mei.json', 'a')
     as file:
52.           file.write(exported)
```

Kode 5.3.12 Main Method Youtube Crawler

Kode 5.3.12 merupakan *main method* dari keseluruhan kode untuk Youtube *Crawler*. Sama seperti crawler lainnya, keseluruhan akun *youtube* didapatkan dari penarikan data melalui *google spreadsheets* yang sebelumnya dibuat.. *Crawler* akan melakukan perulangan sesuai *list* semua akun yang sebelumnya dibuat. Pengecekan terhadap pemerintah daerah yang memliki akun resmi namun tidak memiliki akun influencer dan sebaliknya juga dilakukan pada *main method*. Hasil akhir proses *crawling* ini disimpan dengan nama file youtube_lengkap.json yang berisikan JSON keseluruhan akun pemerintah daerah. Contoh JSON dapat dilihat pada Tabel 4.1.4.

5.4 Pra-Proses Data

5.4.1 *Data Cleaning*

Data yang didapatkan dari hasil crawling masih mengandung berbagai karakter tak bermakna yang dapat mengurangi kualitas data pada proses training. Oleh karena itu dilakukan pembersihan data dengan cara menghilangkan berbagai tanda baca dan simbol tertentu.

```
1. data_komentar['text'] = data_komentar['text'].re
   place('https?:\/\/.*\/\w*', '', regex = True)
2. data_komentar['text'] = data_komentar['text'].re
   place('www.*\/\w*', '', regex = True)
3. data_komentar['text'] = data_komentar['text'].re
   place('\#\w*', '', regex = True)
4. data_komentar['text'] = data_komentar['text'].re
   place('\@\w*', '', regex = True)
```

```
5.  data_komentar['text'] = data_komentar['text'].re
    place('[' + string.punctuation + ']+', ' ', rege
    x = True)
6.  data_komentar['text'] = data_komentar['text'].re
    place('0|[1-9][0-9]*', ' ', regex = True)
7.  data_komentar['text'] = data_komentar['text'].st
    r.replace('[^\w\s#@/:%.,_-
    ]', '', flags = re.UNICODE)
8.  data_komentar['text'] = data_komentar['text'].re
    place('\n', ' ', regex = True)
9.  data_komentar['text'] = data_komentar['text'].re
    place('\s+\s+', ' ', regex = True)
```

Kode 5.4.1 Pemebersihan Data

Kode 5.4.1 merupakan potongan kode pada praproses data menggunakan fungsi regex (regular expression) untuk menghilangkan beberapa karakter yang disebutkan dan menggantinya dengan nilai null atau kosong sehingga data nantinya tidak lagi mengandung karakter tak bermakna.

5.4.2 *Case Folding*

Tahapan *case folding* adalah proses dimana mengubah data ke dalam bentuk huruf kecil, dengan tujuan untuk menyamaratakan format. Adapun potongan kode pada Kode 5.4.2 yang digunakan untuk melakukan *case folding*.

```
1.  data_komentar['text'] = data_komentar['text'].st
    r.lower()
```

Kode 5.4.2 Case Folding ke Lowercase

5.4.3 *Stemming*

Stemming merupakan tahapan untuk mengubah sebuah kalimat menjadi kata dasar. Dalam penelitian ini, stemming dilakukan dengan menggunakan library Sastrawi dengan menggunakan fungsi stem. Adapun Kode 5.4.3 yang digunakan untuk melakukan proses stemming data.

```
1.  factory = StemmerFactory()
```

```
2.  stemmer = factory.create_stemmer()
3.
4.  data_komentar ['text'] = data_komentar ['text'].
    apply(stemmer.stem)
```

Kode 5.4.3 Stemming menggunakan Library Sastrawi

5.4.4 *Tokenizing*

Tokenizing merupakan tahap memecah teks yang dapat berupa kalimat, paragraf atau dokumen, menjadi token-token atau sekumpulan kata. *Tokenizing* memisahkan teks berdasarkan spasi dan menjadikan kata dalam korpus. Untuk melakukan proses *tokenizing*, dalam penelitian ini menggunakan library *pyspark ml* meliputi fungsi *regex tokenizer*. Kode 5.4.4 merupakan potongan kode praproses data untuk melakukan *tokenizing* menggunakan *pyspark ml*. Proses ini merupakan bagian dari proses *spark pipeline*.

```
1.  regexTokenizer = RegexTokenizer(inputCol = "text
    ", outputCol = "tokenized", pattern = "\\W")
```

Kode 5.4.4 Tokenization

5.4.5 *Stopwords Removal*

Stopwords adalah kata umum yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Apabila suatu kata dinilai tidak memiliki makna, maka kata tersebut akan masuk ke dalam daftar kata yang dibuang *stopwords*. Untuk melakukan proses *stopwords removal*, dalam penelitian ini menggunakan library *pyspark ml* meliputi fungsi *StopWordsRemover*. Daftar *stopwords* didapatkan dari penelitian yang sudah dijelaskan pada sub bab 4.3.3. Kode 5.4.5 merupakan potongan kode praproses data untuk melakukan *stopwords removal* menggunakan *pyspark ml*. Proses ini merupakan bagian dari proses *spark pipeline*.

```
1.  f = open("../stopwords-id.txt", "r")
2.  stopwords = f.read().split('\n')
```

```
3.
4.  stopwordsRemover = StopWordsRemover(inputCol = "
    tokenized", outputCol = "stopped").setStopWords(
    stopwords)
```

Kode 5.4.5 Penghilangan Kata yang Tidak Penting

5.5 Pemodelan Random Forest

5.5.1 Pembuatan Model

Dalam tahapan pembuatan model, keselurahan proses akan dijadikan dalam satu *pipeline*. Pembuatan *pipeline* untuk model *random forest* sepenuhnya menggunakan libaray dari pyspark. Sebelum membangun sebuah model, dataset dibagi menjadi 2 bagian data yaitu data training dan data testing. Data training atau data latih digunakan sebagai data untuk pembangunan model. Model inilah yang nantinya digunakan untuk memprediksi kelas dari data testing atau data uji. Model ini kemudian digunakan untuk memprediksi kelas dari data baru yang belum diketahui.

text	category
Ampun bgt mm ko jalan banjar wnsbo. Q k solo jand dr ujung pe ujung banjar jlne lubang tok deg degan bgt. Apa lg 1 minggu sekali pulang pergi solo banjar.	Komplain Layanan Pemerintahan
ampun....heran kok ada pejabat kaya gini...tapi makasih Ya Allah, Indonesia masih diberikan ...Allah Maha Baik	Mengemukakan Pendapat
ampuuun. Tadi dapet info dari warga Nusantara ada beberapa kecelakaan. Karena mereka rata2 belum tau. Harusnya Nusantara jgn dijadikan 1 arah juga.	Berbagi Informasi
Anak anak belum bisa mengurusi kebutuhan keluarga jika menikah muda dan pekerjaan yg tidak mudah dilakukan. Mending ngaji dan sekolah dulu karna kewajiban	Mengemukakan Pendapat
anak saya daftar SMK 1 cimahi hingga per tgl 29 pkl 18,00 DITERIMA di jur, RPL urutan 38 dari 47. tiba2 pkl 20.00 namanya hilang. bayangkan bagaimana perasaan anakku. kalau ga ada solusi terbaik bagi anakku. saya akan lapor ke POLISI perlindungan anak dan HAM dan KPK	Berbagi Informasi
Anak sy pelajar usia 17thn tpi blm py e ktp gmn..apa hrs mengurus ektp atau ttp kartu pelajar	Meminta Informasi
Anak usia berapa mulai bisa ngurus E-ktp,?	Meminta Informasi
Anang Sujana bagus nih walikota bekasi. syar'i	Apresiasi
Ancer2 dari giwangan naik apa y..mksih mohon infonya	Meminta Informasi
anda gak salah gan,,,,,permasalahhan nya markupini loh,,,,kalo semen hanya butuh 10 sak,,,,mereka jadikan 20 sak,,,,itu permasalahannya,,,,	Mengemukakan Pendapat

Gambar 5.5.1 Contoh Dataset

Gambar 5.5.1 adalah contoh dari dataset yang digunakan. Data yang digunakan sebagai data training dan data test telah sama-sama melalui tahapan praproses data.

```
1.  (trainingData, testData) = data.randomSplit([0.7
    , 0.3], seed = 100)
2.
3.  vocab_size = <angka>
4.  min_DF = <angka>
5.  countVectors = CountVectorizer(inputCol = "stopp
    ed", outputCol = "features", vocabSize = vocab_s
    ize, min_DF = min_DF)
6.
7.  num_trees = <angka>
8.  max_depth = <angka>
9.  rfModel = RandomForestClassifier(
10.         labelCol = "label", \
11.         featuresCol = "features", \
12.         numTrees = num_trees, \
13.         maxDepth = max_depth)
14. pipeline = Pipeline(stages = early_stages + [rfM
    odel, labelConverter])
15. rfPipeline = pipeline.fit(trainingData)
16. predictions = rfPipeline.transform(testData)

17. rfPipeline.save('rfPipeline')
```

Kode 5.5.1 Pembentukan Model

Baris 1 pada Kode 5.5.1 merupakan potongan kode yang digunakan untuk membagi dataset menjadi data training dan data test. Pembagian data training dan data test dalam penelitian ini adalah 70:30.

Baris 3-5 pada Kode 5.5.1 merupakan potongan kode yang digunakan untuk mengubah teks dari data komentar menjadi sebuah *vector*. Hal ini perlu dilakukan karena untuk mengetahui pola teks, mesin tidak dapat membaca teks melainkan angka. Proses ini juga merupakan bagian dari *spark pipeline*.

Baris 7-13 Kode 5.5.1 merupakan potongan kode yang digunakan untuk pembuatan model. Dalam pembentukan model klasifikasi *random forest*, diperlukan sebuah parameter, yaitu berupa *num trees dan max depth*. Penentuan parameter disini dimaksudkan untuk mencari nilai akurasi yang optimal. Semakin besar nilai akurasi menunjukkan bahwa model yang

dibentuk semakin baik. Tahapan ini adalah proses terpenting dalam *pipeline.*

Baris 14 Kode 5.5.1 merupakan potongan kode yang digunakan untuk pembuatan keseluruhan *pipeline* dari hasil penggabungan *stages* awal yang berisikan proses *tokenizing*, *stopwords removal*, *countVectors*, *label_stringIdx* yang berfungsi untuk mengubah label ke dalam bentuk indeks angka, proeses klasifiikasi dan *labelConverter* yang digunakan untuk mengembalikan label dalam bentuk *string*.

Baris 15-16 Kode 5.5.1 merupakan potongan kode yang digunakan untuk melakukan *method fit*, yang digunakan untuk mempelajari parameter model dari data training, dan *method transform* yang menerapkan model transformasi ini ke data test.

Baris 17 Kode 5.5.1 merupakan potongan kode yang digunakan untuk menyimpan model klasifikasi *random forest*. Model klasifikasi disini adalah *pipeline* yang sudah dibangun sebelumnya. Model yang disimpan adalah model dengan akurasi yang paling baik untuk digunakan kembali pada saat proses klasifikasi data *stream.*

5.5.2 Uji Coba Pemodelan Klasifikasi menggunakan *Random Forest*

Tahap uji coba pemodelan klasifikasi menggunakan *Random Forest* merupakan tahapan yang dilakukan untuk membentuk model topik terbaik dengan melakukan uji coba sesuai parameter, yakni *num of trees* dan *max depth*. Parameter minimum *leaf size* tidak digunakan di penelitian ini dikarenakan spark tidak menyediakan fungsi *tuning* parameter tersebut.

1. Penentuan *Num of Trees*

   Pada metode *Random Forest*, *num of trees* merupakan banyak pohon keputusan yang dibuat dalam pembentukan model. Penentuan banyak pohoh keputusan merupakan tahap yang penting dalam menentukan model, guna menghasilkan model yang terbaik. Jumlah pohon keputusan yang terbaik bergantung pada dataset yang digunakan,

sehingga jumlah pohon keputusan yang terbaik setiap dataset berbeda. Penentuan jumlah pohon keputusan diawali dengan memberikan nilai sebesar 10, pembulatan dari nilai 9 agar interval lebih mudah berpola berdasarkan penelitian yang dilakukan oleh sebelumnya [6]. Jika hasil akurasi masih belum mendapatkan nila yang baik, maka percobaan penambahan jumlah pohon keputusan akan dilakukan. Interval yang digunakan dalam penelitian ini adalah 10 sampai menemukan nilai akurasi terbaik. Percobaan ini akan dilakukan jika sudah menemukan nilai akurasi terbaik dari percobaan pada parameter *max depth*.. Berdasarkan hasil uji coba jumlah pohon keputusan, nilai akurasi yang muncul akan dicatat untuk dianalisis grafiknya secara visual, dan dilakukan penghitungan nilai *precision, recall*, dan *f-measure*. Adapun pengkodean yang digunakan untuk penentuan jumlah iterasi ditunjukkan dengan Kode 5.5.2.

```
num_trees = <angka>
max_depth = <angka>

rfModel = RandomForestClassifier(
        labelCol = "label", \
        featuresCol = "features", \
        numTrees = num_trees, \
        maxDepth = max_depth)
```

Kode 5.5.2 *Tuning* Parameter Banyak Pohon Keputusan

2. Penentuan *Max Depth*

Pada penelitian ini, uji coba penentuan *max depth* atau maksimum kedalaman pohon dilakukan untuk menemukan nilai akurasi terbaik. Uji coba *max depth* juga merupakan tahap yang penting dalam menentukan model, hal ini untuk menghasilkan model yang terbaik, model dapat dikatakan terbaik apabila model memiliki nilai akurasi yang tinggi, semakin tinggi nilai akurasi, menunjukkan akurasi model yang semakin baik. Penentuan maksimum kedalaman pohon diawali dengan memberikan nilai sebesar 16

berdasarkan penelitian yang dilakukan oleh sebelumnya [6]. Uji coba penambahan maksimum kedalaman pohon penentuan dengan interval sebesar 2 sampai maksimum nilai yang dapat diterima oleh *spark* yaitu 30. Berdasarkan eksperimen *max depth*, nilai akurasi yang muncul akan dicatat untuk dianalisis tren nilainya secara visual dan dilakukan penghitungan *precision, recall*, dan *f-measure*. Adapun pengkodean yang digunakan untuk penentuan jumlah topik ditunjukkan dengan Kode 5.5.3.

```
num_trees = <angka>
max_depth = <angka>

rfModel = RandomForestClassifier(
        labelCol = "label", \
        featuresCol = "features", \
        numTrees = num_trees, \
        maxDepth = max_depth)
```

Kode 5.5.3 *Tuning* Parameter Maksimum Kedalaman Pohon

Jika akurasi yang didapatkan belum mendapatkan nilai yang begitu baik, maka akan dilakukan penyeimbangan dataset. Dataset akan dibuat seimbang setiap labelnya untuk *data training* agar mendapatkan nilai akurasi yang baik. Hal ini perlu dilakukan karena metode *random forest* tidak begitu bagus menangani data yang tidak seimbang.

5.5.3 Validasi Model

Validasi model klasifikasi disini mengacu pada nilai akurasi yang didapatkan dari hasil uji coba parameter. Model dapat dikatakan terbaik jika memiliki nilai akurasi yang tinggi. Nilai perplexity akan dicatat untuk dianalisis tren nilainya secara visual. Nilai *preicision, recall* dan *f-measure* juga dijadikan perbandingan pada penilitian ini. Sehingga pada akhirnya parameter yang dipilih adalah yang memiliki nilai rata-rata paling tinggi.

```
print("Accuracy = %s" % metrics.accuracy)
```

```
labels = rdd.map(lambda lp: lp.label).distinct()
.collect()
for label in sorted(labels):
   print("Class %s precision = %s" % (label, met
rics.precision(label)))
   print("Class %s recall = %s" % (label, metric
s.recall(label)))
   print("Class %s F1 Measure = %s" % (label, me
trics.fMeasure(label, beta = 1.0)))
```

Kode 5.5.4 Membuat MultiClass Metrics untuk Validasi Akurasi Model

Kode 5.5.4 merupakan potongan kode yang digunakan untuk melakukan validasi terhadap model yang sudah dibuat sebelumnya. Sebelum dilakukan validasi, bentuk dari hasil *testing* diubah terlebih dahulu kedalam bentuk *rdd*. Nilai yang akan dihitung dalam tahap ini meliputi akurasi model itu sendiri serta *precision, recall dan f-measure* pada setiap label menggunakan *multi class metrics* pada *pyspark*.

5.6 *Data Streamer*

5.6.1 Kafka

Facebook Streamer

Pembuatan *streamer* untuk Facebook hampir sama dengan *crawler* karena Facebook tidak menyediakan *API* untuk melakukan *stream* data. Proses *stream* data dilakukan dengan cara melakukan perulangan sistem *crawler* yang dapat dilihat pada sub bab 5.3.2. Setelah *crawler* pada akun terakhir selesai maka *crawler* akan melakukan crawling pada akun pertama. Sebelum data diteruskan menuju *kafka*, perlu dilakukannya *parsing* pada data JSON agar mendapatkan data yang dibutuhkan saja yaitu komentar.

```
def getCommentDocument(self, complete_dict):
        commentdocument = complete_dict['comment
']
        for comment in commentdocument:
            comment['_id'] = comment[self.commen
t_id_key]
```

```
5.              comment.pop(self.comment_id_key, Non
    e)
6.              comment[self.account_id_key] = compl
    ete_dict['account'][self.account_id_key].lower()

7.          return commentdocument
```

Kode 5.6.1 *Parsing* data Komentar *Facebook*

Kode 5.6.1 menunjukkan isi kode dari *method* getCommentDocument dimana untuk mengambil komentar dari data satuan *posting* Facebook.

```
1.  def send_message(self, dict_or_json):
2.
3.      producer = self.topic.get_producer(
4.          serializer=self.serializer_method,
5.          sync=True,
6.          max_request_size=1024 * 1024 * 10,
7.          use_rdkafka=True,
8.          delivery_reports=False,
9.      )
10.
11.     with producer as producer:
12.         try:
13.             producer.produce(dict_or_json)
14.         except Exception as e:
15.             logging.warning(e)
16.
17.     latest_available_offsets = json.dumps(self.t
    opic.latest_available_offsets())
18.     logging.debug('Latest available offsets: ' +
     latest_available_offsets)
```

Kode 5.6.2 Mengirimkan Data ke *Kafka*

Kode 5.6.2 menunjukkan isi kode dari *method* send_message dimana untuk mengirimkan data komentar ke *kafka* yang sebelumnya sudah di-*parsing*.

```
1.  def pushCommentDocument(self, complete_dict):
2.          comment_document = self.yp.getCommentDoc
    ument(complete_dict)
3.          for comment in comment_document:
```

```
4.                 self.ymc.updateComment(comment_docum
    ent)
5.                 self.ykc.send_message(comment)
```

Kode 5.6.3 Proses Data dari *Crawler*

Kode 5.6.3 menunjukkan isi kode dari *method* pushCommentDocument dimana pada *method* ini dilakukan pemanggilan *method* yang dijelaskan sebelumnya. Hasil data *parsing* berupa *list* komentar. Data akan dikirimkan satu-persatu dengan melakukan proses perulangan pada data *list* komentar.

Twitter Streamer

Pembuatan *streamer* untuk *Twitter* berbeda dengan *Facebook* karena *Twitter* menyediakan *API* untuk melakukan proses *streaming* data. Namun untuk menyalurkan data juga menggunakan *kafka* sebagai data *pipeline*. Berbeda dengan *crawler*, proses ini mengambil data akun-akun pemerintah dan *influencer* melalui database secara langsung.

Proses pengambilan data yaitu t*weet* akan masuk berdasarkan *tweet* balasan yang dikirimkan netizen kepada pemerintahan atau *influencer*. Data *tweet* akan disalurkan ke salah satu topik kafka. Adapun *class* dan *method* pada sistem *twitter stream* ini akan dijelaskan sebagai berikut..

```
1.  class StdOutListener(StreamListener):
2.      def on_data(self, data):
3.          producer.send_messages("twitter-stream-
    reply", data.encode('utf-8'))
4.          print(data)
5.          return True
6.
7.      def on_error(self, status):
8.          print(status)
```

Kode 5.6.4 Twitter Stream Listener

Kode 5.6.4 menunjukkan isi kode dari *class* StdOutListener dimana pada streamer ini berfungsi sebagai penerus data ke

kafka. Pada *class* ini juga dibuat kode untuk pengecekan data error yang masuk.

```
1.  def ambil_list():
2.      lists = collection_pemda.find({})
3.      for pemda in list(lists):
4.          if pemda['twitter_resmi'] != '':
5.              list_accountID.append(pemda['twitter
    _resmi'])
6.          if pemda['twitter_influencer'] != '':

7.              list_accountID.append(pemda['twitter
    _influencer'])
8.
9.      return list_accountID
```

Kode 5.6.5 Mengambil List Akun Twitter

Kode 5.6.5 menunjukkan isi kode dari *method* ambil_list dimana untuk pengambilan daftar akun-akun *twitter* yang terdapat di database. Akun-akun yang dimasukkan ke dalam *list* adalah akun resmi pemerintahan dan *influencer*.

```
1.  if __name__ == '__main__':
2.      #Twitter token
3.      access_token = <Twitter Access Token>
4.      access_token_secret = <Twitter Access Token
    Secret>
5.      consumer_key = <Twitter Consumer Key>
6.      consumer_secret = <Twitter Consumer
    Secret>
7.
8.      #Mongo Client
9.      client = pymongo.MongoClient()
10.     database = client['egovbench']
11.     collection = database['list_pemda']
12.
13.     #Setup kafka client
14.     kafka = KafkaClient("localhost:9092")
15.     #Kafka simple producer
16.     producer = SimpleProducer(kafka)
17.     l = StdOutListener()
18.     #Twitter auth
```

```
    auth = OAuthHandler(consumer_key, consumer_s
ecret)
    auth.set_access_token(access_token, access_t
oken_secret)
    #Create stream api
    stream = Stream(auth, l, wait_on_rate_limit=
True, wait_on_rate_limit_notify=True)

    while True:
        try:
            stream.filter(track=ambil_list())
        except Exception as e:
            print("type error: " + str(e))
```

Kode 5.6.6 Main Method Twitter Streaming

Kode 5.6.6 merupakan *main method* dari keseluruhan kode untuk Twitter *Streamer*. Akun keseluruhan akan dimuat dalam bentuk *list* diamana data akun Twitter didapatkan *method* ambil_list. Dalam method ini sama seperti *crawler* juga menghubungkan *client* dengan Twitter *API* menggunakan *OAuthHandler* yang dibantu dengan menggunakan *library tweepy*. Proses *streamer* ini akan dijalankan terus menerus dengan adanya pengecekan jika terjadi error maka *streamer* akan menjalankan lagi pengambilan data.. Data yang dikirimkan oleh proses *stream* ini berupa JSON dari setiap *tweet* yang masuk.

Youtube Streamer

Pembuatan *streamer* untuk Youtube sama dengan Facebook karena Youtube tidak menyediakan *API* untuk melakukan proses *streaming* data. Proses. Proses *stream* data dilakukan dengan cara melakukan perulangan sistem *crawler* yang dapat dilihat pada sub bab 5.3.4. Penggunaan *method* untuk stream dapat dilihat pada Kode 5.6.1, Kode 5.6.2, dan Kode 5.6.3.

5.6.2 *Spark Streaming*

Tahapan ini adalah tahapan setelah data *stream* masuk dari ke *kafka*. Data dari *kafka* akan ditarik berdasarkan topik stream per media sosial. *Spark streaming* akan melakukan proses terhadap

data yang masuk. Proses yang akan dilakukan pada data akan dijalankan per-*batch* dimana pada proses ini dilakukan setiap 10 detik sekali.

```
1.  def parsing_facebookComment(comment):
2.      comment_id = comment['_id']
3.      comment_createdDate = comment['comment_creat
    edDate']
4.      page_id = comment['page_id']
5.      text = comment['comment_message']
6.
7.      return comment_id, comment_createdDate, page
    _id, text
```

Kode 5.6.7 Parsing Data Facebook yang dibutuhkan

Kode 5.6.7 menunjukkan isi kode dari *method* parsing_facebookComment dimana untuk melakukan *parsing* terhadap data komentar yang berasal dari media sosial Facebook. Data yang *parsing* adalah data yang diperlukan untuk melakukan klasifikasi. *Method* parsing_youtubeComment untuk melakukan *parsing* terhadap komentar dari *youtube* mempunyai kode yang sama dengan Facebook, hanya saja mempunyai perbedaan pada inisiasi variabel

```
1.  def parsing_tweetReply(tweet):
2.      tweet_id = ''
3.      tweet_createdDate = ''
4.      account_id = ''
5.      text = ''
6.
7.      if tweet['in_reply_to_screen_name'] is not N
    one:
8.          if tweet['in_reply_to_screen_name'] in l
    ists:
9.              if not tweet['truncated']:
10.                 text = tweet['text']
11.
12.             elif tweet['truncated']:
13.                 text = tweet['extended_tweet']['
    full_text']
```

```
14.
15.             tweet_id = tweet['id_str']
16.             tweet_createdDate = eu.formatTwitter
    Time(tweet['created_at'])
17.             account_id = tweet['in_reply_to_scre
    en_name'].lower()
18.
19.     return tweet_id, tweet_createdDate, account_
    id, text
```

Kode 5.6.8 Parsing Data Twitter yang dibutuhkan

Kode 5.6.8 menunjukkan isi kode dari *method* parsing_tweetReply dimana untuk melakukan *parsing* terhadap data komentar yang berasal dari media sosial *twitter*. Sedikit berbeda dengan Facebook dan Youtube data dari *twitter* perlu dilakukan pengecekan pada isi komentar apakah terpotong atau tidak.

```
1.  def parsing_youtubeComment(comment):
2.      comment_id = comment['_id']
3.      comment_createdDate = comment['comment_creat
    edDate']
4.      channel_id = comment['channel_id']
5.      text = comment['comment_message']
6.
7.      return comment_id, comment_createdDate, chan
    nel_id, text
```

Kode 5.6.9 Parsing Data Youtube yang dibutuhkan

Kode 5.6.9 menunjukkan isi kode dari *method* parsing_youtubeComment dimana untuk melakukan *parsing* terhadap data komentar yang berasal dari media sosial Facebook. Proses ini sama seperti Facebook, hanya saja data masuk dari *kafka* sedikit berbeda.

```
1.  def pd_to_sparkDF(df):
2.      df = spark.createDataFrame(df)
3.      return df
4.
5.  def sparkDF_to_pd(df):
6.      df = df.toPandas()
```

```
7.      df = df[df.text != '']
8.      return df
```

Kode 5.6.10 Mengubah ke Pandas Dataframe atau Spark Dataframe

Kode 5.6.10 menunjukkan isi kode dari *method* pd_to_sparkDF dan sparkDF_to_pd dimana untuk melakukan transformasi data ke *pandas dataframe* maupun ke *sprak dataframe*. Hal ini dilakukan karena untuk proses *data cleaning* dan *stemming* tidak dapat dilakukan pada *spark datataframe*. Sedangkan untuk proses klasifikasi harus menggunakan *spark dataframe.*

```
1.  def processFacebook(rdd):
2.      if rdd.isEmpty():
3.          print('No Data Received from Facebook\n'
    )
4.      else:
5.          schema = StructType( [
6.              StructField('_id', StringType()),
7.              StructField('comment_createdDate', S
    tringType()),
8.              StructField('page_id', StringType())
    ,
9.              StructField('text', StringType()),
10.             ])
11.         df = spark.createDataFrame(rdd, schema)
12.         df = sparkDF_to_pd(df)
13.         process_item(df)
```

Kode 5.6.11 Proses RDD Facebook

Kode 5.6.11 menunjukkan isi kode dari *method* processFacebook dimana untuk melakukan proses pengecekan *rdd* dan sekaligus pembuatan skema *spark dataframe* dari data *rdd* yang sudah melewati proses *parsing* dari media sosial Facebook. *Method* processTwitter dan processYoutube untuk memproses data dari Twitter dan Youtube mempunyai kode yang sama hanya saja structField yang sedikit berbeda.

5.7 Klasifikasi

5.7.1 Pra-proses Data Stream

Tahapan ini mempunyai proses yang sama dengan pra-proses data pada pembuatan model. Data masuk akan dilakukan proses *data cleaning*, *case folding*, *stemming*, *tokenizing*, dan *stopwords removal*. Untuk 3 pra-proses pertama dapat dilihat pada Kode 5.4.1, Kode 5.4.2, dan Kode 5.4.3. Namun sedikit berbeda dengan proses sebelumnya, proses *tokenizing* dan *stopwords removal* sudah menjadi satu bagian dengan proses selanjutnya pada *spark pipeline* yang dapat dilihat pada Kode 5.4.4 dan Kode 5.4.5.

5.7.2 Data Klasifikasi

Data yang sudah melewati pra-proses akan dilanjutkan dengan melakukan klasifikasi terhadap data. Model yang sebelumnya sudah dibuat akan dimuat dalam proses ini.

```
1.  def process_item(df):
2.      df = preprocess(df)
3.      if df.empty:
4.          print('No Data Received from Twitter\n')
5.          gc.collect()
6.      else:
7.          df = pd_to_sparkDF(df)
8.          df = rf_pipeline(df)
9.          json_result = json_parsing(df)
10.         json_categorized = categorization(json_r
    esult)
11.         gc.collect()
```

Kode 5.7.1 Proses Pembentukan Dataset

Kode 5.7.1 menunjukkan isi kode dari *method* process_item dimana pada *method* ini dilakuakan pengecekan terhadap data dan pemanggilan *method* lainnya seperti *preprocess*, transformasi *dataframe*, *parsing* JSON dan *garbage collection* untuk membuang data yang sudah tidak diperlukan pada memori.

```
def rf_pipeline(dataset):
    predict = rfPipeline.transform(dataset)

    drop_list = ['tokenized', 'stopped', 'features', 'rawPrediction', 'probability', 'prediction']
    data = predict.select([column for column in predict.columns if column not in drop_list])
    df = sparkDF_to_pd(data)
    if 'account_id' in df.columns:
        df.rename(columns={'text':'reply_message'}, inplace=True)
    else:
        df.rename(columns={'text':'comment_message'}, inplace=True)

    return df
```

Kode 5.7.2 Proses Klasifikasi Data Stream

Kode 5.7.2 menunjukkan isi kode dari *method* rf_pipeline dimana untuk melakukan klasfikasi data. Setelah data terklasifikasi maka diperlukan pembersihan data yang tidak diperlukan sehingga data siap untuk masuk ke database.

5.7.3 Menambahkan ke Database

Data yang sudah melewati proses klasfikasi akan dilanjutkan dengan menambahkan data hasil ke dalam database. Data hasil klasfikasi dimasukkan database berdasarkan media media sosial masing-masing sumber data.

```
def json_parsing(df): # Pandas
    json_result = df.to_json(orient='records')
    json_result = json.loads(json_result)

    return json_result
```

Kode 5.7.3 Parsing Pandas Dataframe ke JSON

Kode 5.7.3 menunjukkan isi kode dari *method* json_parsing dimana untuk melakukan parsing data hasil ke bentuk JSON. Hal ini dilakukan untuk memperudah pemasukan data ke dalam

database karena dilakukan pengecekan sumber data media sosial.

```
def insert_mongo(json_categorized):

    if 'account_id' in json_categorized:
        twMongo.updateComment(json_categorized)

    elif 'page_id' in json_categorized:
        fbMongo.updateComment(json_categorized)

    elif 'channel_id' in json_category:
        ytMongo.updateComment(json_categorized)
```

Kode 5.7.4 Memasukkan Data Hasil Klasifikasi ke DB

Kode 5.7.4 menunjukkan isi kode dari *method* insert_mongo dimana untuk memasukkan data klasifikasi ke dalam database. Data akan dimasukkan ke dalam database berdasarkan media sosial.

```
def updateComment(self, commentDocument):
        try:
            self.comment_collection.update_one(
                {
                    '_id': commentDocument['_id']
]
                },
                {
                    '$set': commentDocument
                },
                upsert=True
            )
        except Exception as e:
            logging.warning(e)
```

Kode 5.7.5 Query Upsert ke MongoDB

Kode 5.7.5 menunjukkan isi kode dari *method* updateComment dimana untuk melakukan query ke database MongoDB. Query yang digunakan adalah *upsert* untuk sekaligus melakukan pengecekan terhadap data agar tidak terduplikasi. Jika data

sudah ada dalam database maka dilakukan *update* dan jika belum maka melakukan query *insert*.

5.8 Visualisasi

Visualisasi data dalam penelitian ini merupakan pembuatan grafik dan statistik harian yang dihasilkan dari data komentar netizen pada media sosial pemerintahan yang masuk melalui aplikasi menggunakan proses *streaming*. Ada beberapa grafik yang digunakan pada penelitian ini yaitu diagram batang untuk melihat data tertinggi setiap label klasifikasi dan untuk memperlihatkan nilai standar deviasi komentar, diagram garis untuk menampilkan perkembangan komentar netizen yang masuk ke media sosial pemerintah daerah. Visualisasi grafik ini menggunakan *library Highcarts* yang menyediakan banyak jenis grafik.

```
1.  date_default_timezone_set("Asia/Jakarta");
2.  $pemda = listPemdaModel::all()->count();
3.  $facebook_resmi = listPemdaModel::where('faceboo
    k_resmi', '!=', "")->count();
4.  $facebook_influencer = listPemdaModel::where('fa
    cebook_influencer', '!=', "")->count();
5.  $twitter_resmi = listPemdaModel::where('twitter_
    resmi', '!=', "")->count();
6.  $twitter_influencer = listPemdaModel::where('twi
    tter_influencer', '!=', "")->count();
7.  $youtube_resmi = listPemdaModel::where('youtube_
    resmi', '!=', "")->count();
8.  $youtube_influencer = listPemdaModel::where('you
    tube_influencer', '!=', "")->count();
9.  $komentar = facebookCommentsModel::where('commen
    t_createdDate', date("Y-m-d"))-
    >count() + twitter_replyModel::where('tweet_crea
    tedDate', date("Y-m-d"))-
    >count() + youtubeCommentsModel::where('comment_
    createdDate', date("Y-m-d"))->count();
10. $semuaKomentar = facebookCommentsModel::all()-
    >count() + twitter_replyModel::all()-
    >count() + youtubeCommentsModel::all()-
    >count();
```

```
11. $rataKomentar = round($semuaKomentar / $pemda,2)
    ;
```

Kode 5.8.1 *Query* Status Harian Komentar

Kode 5.8.1 merupakan potongan kode dari *controller* untuk beranda dimana untuk melakukan *query* ke MongoDB untuk mendapatkan nilai status harian komentar. Pada variabel semuaKomentar menunjukkan total nilai data komentar yang berhasil diklasifikasikan. Serta pada *query* ini menunjukkan banyak akun resmi pemerintah daerah maupun *influencer* yang aktif per tanggal hari ini. Selain itu, rata-rata komentar dalam seluruh pemda juga didefinisikan pada fungsi ini. Tampilan statistik dapat dilihat pada Gambar 5.8.1.

```
1.  $pemda = listPemdaModel::where('_id', (int)$id)-
    >first();
2.
3.  $labelKlasifikasi = ['Berbagi Informasi', 'Memin
    ta Informasi', 'Mengemukakan Pendapat', 'Apresia
    si', 'Komplain Pelayanan'];
4.
5.  $komentarFacebookResmi = facebookCommentsModel::
    where('page_id', $pemda->facebook_resmi)-
    >get();
6.
7.  foreach($labelKlasifikasi as $lk) {
8.    $jumlahKomentarFacebookResmi[] = $komentarFace
    bookResmi->where('class', $lk)->count();
9.  }
10.
11. $komentarFacebookInfluencer = facebookCommentsMo
    del::where('page_id', $pemda-
    >facebook_influencer)->get();
12.
13. foreach($labelKlasifikasi as $lk) {
14.   $jumlahKomentarFacebookInfluencer[] = $komenta
    rFacebookInfluencer->where('class', $lk)-
    >count();
15. }
16.
17. $komentarTwitterResmi = twitter_replyModel::wher
    e('account_id', $pemda->twitter_resmi)->get();
```

```
18.
19. foreach($labelKlasifikasi as $lk) {
20.   $jumlahKomentarTwitterResmi[] = $komentarTwitt
    erResmi->where('class', $lk)->count();
21. }
22.
23. $komentarTwitterInfluencer = twitter_replyModel:
    :where('account_id', $pemda-
    >twitter_influencer)->get();
24.
25. foreach($labelKlasifikasi as $lk) {
26.   $jumlahKomentarTwitterInfluencer[] = $komentar
    TwitterInfluencer->where('class', $lk)-
    >count();
27. }
28.
29. $komentarYoutubeResmi = youtubeCommentsModel::wh
    ere('channel_id', $pemda->youtube_resmi)-
    >get();
30.
31. foreach($labelKlasifikasi as $lk) {
32.   $jumlahKomentarYoutubeResmi[] = $komentarYoutu
    beResmi->where('class', $lk)->count();
33. }
34.
35. $komentarYoutubeInfluencer = youtubeCommentsMode
    l::where('channel_id', $pemda-
    >youtube_influencer)->get();
36.
37. foreach($labelKlasifikasi as $lk) {
38.   $jumlahKomentarYoutubeInfluencer[] = $komentar
    YoutubeInfluencer->where('class', $lk)-
    >count();
39. }
```

Kode 5.8.2 *Query* MongoDB untuk Viusalisasi Klasifikasi

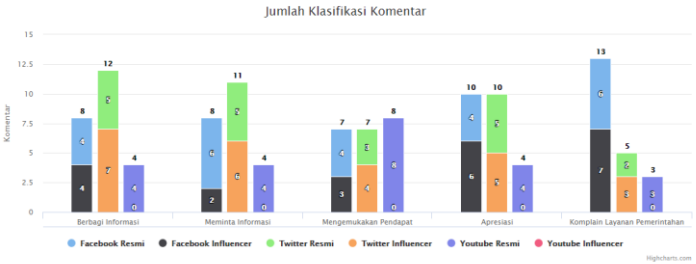
Kode 5.8.2 merupakan potongan kode dari *controller* untuk klasifikasi dimana untuk melakukan *query* ke MongoDB untuk mendapatkan nilai total komentar terklasifikasi per label setiap satuan pemerintah daerah. Data yang diambil adalah data komentar dari semua media sosial resmi maupun *influencer*. Tampilan grafik dapat dilihat pada Gambar 5.8.2.

```
1.  for($i = 9; $i >= 0; $i --) {
2.   $tanggal[] = date('Y-m-d', strtotime('-
    '.$i.' days',  strtotime(date("Y-m-d"))));
3.  }
4.
5.  foreach ($tanggal as $tgl) {
6.   $hitungKomenFBResmi[] = facebookCommentsModel::
    where('comment_createdDate', $tgl)-
    >where('page_id', $pemda->facebook_resmi)-
    >count();
7.   $hitungKomenTWResmi[] = twitter_replyModel::whe
    re('tweet_createdDate', $tgl)-
    >where('account_id', $pemda->twitter_resmi)-
    >count();
8.   $hitungKomenYTResmi[] = youtubeCommentsModel::w
    here('comment_createdDate', $tgl)-
    >where('channel_id', $pemda->youtube_resmi)-
    >count();
9.   $hitungKomenFBInfluencer[] = facebookCommentsMo
    del::where('comment_createdDate', $tgl)-
    >where('page_id', $pemda->facebook_influencer)-
    >count();
10.  $hitungKomenTWInfluencer[] = twitter_replyModel
    ::where('tweet_createdDate', $tgl)-
    >where('account_id', $pemda-
    >twitter_influencer)->count();
11.  $hitungKomenYTInfluencer[] = youtubeCommentsMod
    el::where('comment_createdDate', $tgl)-
    >where('channel_id', $pemda-
    >youtube_influencer)->count();
12. }
```

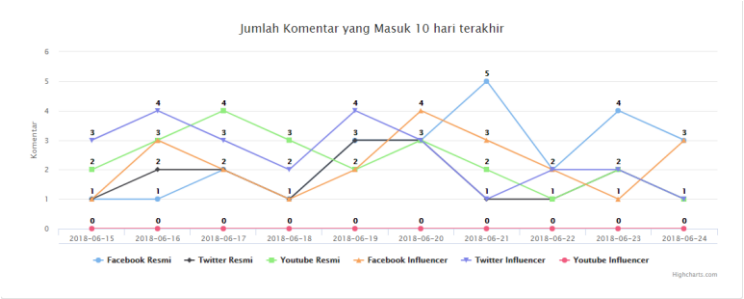
Kode 5.8.3 *Query* MongoDB untuk Pergerakan Komentar Masuk

Kode 5.8.3 merupakan potongan kode dari *controller* untuk klasifikasi dimana untuk melakukan *query* ke MongoDB untuk mendapatkan nilai total komentar per harinya setiap satuan pemerintah daerah. Data yang diambil adalah 10 hari terakhir data komentar gabungan komentar dari akun resmi dan *influencer* dari setiap media sosial. Tampilan grafik dapat dilihat pada Gambar 5.8.3.

Gambar 5.8.1 ***Daily Statistics***



Gambar 5.8.2 Jumlah Klasifikasi Komentar per Pemda



Gambar 5.8.3 Jumlah Komentar yang masuk 10 hari terakhir

Halaman ini sengaja dikosongkan

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan dijelaskan hasil dan pembahasan terkait analisa dan pengujian aplikasi yang meliputi tiga hal, yaitu analisa hasil model klasifikasi, pengujian fungsional dan non fungsional.

6.1 Analisis Hasil Model Klasifikasi

Pada analisis hasil model klasifikasi akan membahas mengenai hasil pengujian model menggunakan perhitungan akurasi, *precision*, *recall*, dan *f-measure* dari masing-masing klasifikasi yang dilakukan. Adapun langkah – langkah yang dilakukan dijelaskan pada hal berikut.

6.1.1 Memuat Data

Total data yang berhasil didapatkan dari hasil *crawling* berjumlah 1400396 data dari media sosial Facebook, Twitter, and Youtube. Detail data yang berhasil terakuisisi sudah dijelaskan pada sub bab 5.1 serta jumlah data yang diberi label. Adapun detail jumlah data per label dari setiap sosial media ditunjukkan pada Tabel 6.1.1.

Tabel 6.1.1 Distribusi Label per Sosial Media

Label	Facebook	Twitter	Youtube
Berbagi Informasi	590	340	78
Meminta Informasi	863	154	112
Mengemukakan Pendapat	26	85	1422
Melaporkan Permasalahan Layanan	118	31	23
Apresiasi	661	77	373
Komplain	294	74	189

Meminta Peningkatan Layanan	75	18	102
Meminta Layanan Baru	23	9	49
Memasarkan Acara	4	57	5
Pembentukan Komunitas atau Identitas	12	10	33
Total	2666	855	2386

6.1.2 Pra-proses Data

Pendefinisian Data Uji

Dalam penelitian ini untuk membuat model berdasarkan data komentar netizen pada media sosial pemerintah daerah, digunakan data hanya 5907 untuk dilabeli dari total data 1400396. Data akhir yang yang digunakan untuk data uji adalah 5786 dari proses penggabungan dan penghilangan label yang sudah dijelaskan di sub bab 5.1.

Hasil Data Setelah Pra-Proses

Data awal yang terdiri dari 5786 data yang mempunyai kata sebanyak 163923 berubah menjadi 103413 kata setelah dilakukan beberapa tahapan pra-proses pada sub bab 4.3.3. Hal ini terjadi dikarenakan pada saat melewati tahapan *data cleaning* dan *stopwords removal* sehingga kata yang memiliki karakter dan kata yang tidak penting akan terhapus dan setiap data. Pada tahapan ini data yang berhasil dihasilkan dapat dilihat pada Gambar 6.1.1.

['admin', 'izin', 'kunjung', 'individu', 'rekreasi', 'keluarga', 'prosedur', 'rombong', 'sekolah', 'langsung', 'masuk', 'terimakasih']
['admin', 'hormat', 'terima', 'kasih', 'vidio', 'tungu', 'heheh', 'tingkat', 'gan', 'kualitas', 'heheh', 'peace']
['aduuuhh', 'admin', 'kerja', 'ya', 'kalo', 'adil', 'post', 'skolah', 'skolah', 'patok', 'kalo', 'skolah', 'admin', 'jatuh', 'skolah', 'mikir', 'kalo', 'ngepost', 'nama', 'solok', 'selatan']
['warga', 'cekam', 'takut', 'polisi', 'gerak', 'cepat']
['aher', 'bangun', 'jokowi', 'tinggal', 'resmi', 'bangun', 'bandar', 'udara', 'apbn', 'perintah', 'pusat', 'sehat', 'pa', 'aher', 'kerja', 'layak', 'presiden']
['ahmad', 'arif', 'fachrudin', 'dengar', 'ss', 'mengupdate', 'informasi', 'tugas', 'dinas', 'sosial', 'kota', 'surabaya', 'jemput', 'nenek', 'suminah', 'tampung', 'liponsos', 'terima', 'kasih', 'bantu', 'kawan', 'odp', 'rt']
['ahok', 'orang', 'hebat', 'yg', 'tinggal', 'indonesia', 'bangun', 'bangsa', 'coba', 'kalo', 'orang', 'yg', 'pinter', 'pinter', 'negri', 'terima', 'dg', 'layak', 'ahok', 'indonesia', 'maju']
['ahok', 'kalo', 'bilang', 'mari', 'debat', 'ambil', 'contoh', 'bangun', 'kalo', 'bangun', 'pake', 'uang', 'periksa', 'kerja', 'proses', 'kpk', 'jaksa', 'proses', 'buka', 'koreng', 'kalo', 'buka', 'koreng', 'mari', 'jakarta', 'waduhh', 'gimana', 'tuh', 'ya', 'p', 'ahok', 'bilang', 'periksa', 'kerja', 'proses', 'kpk', 'jaksa', 'proses']
['air', 'mata', 'buaya', 'kau', 'menang', 'lg', 'lihat', 'infra', 'struktur', 'jalan', 'simalungun']
['air', 'cepet', 'surut', 'mantap', 'kerja', 'punggawa', 'dki']

Gambar 6.1.1 Hasil Pra-Proses

Sehingga dapat dilihat pada Tabel 6.1.2 jumlah kata pada data yang berkurang setiap tahapan pra-prosesnya.

Tabel 6.1.2 Hasil Tiap Tahapan Pra-Proses

Tahapan	Jumlah Kata pada Data
Data Awal	163923
Data Cleaning	164405
Case Folding	164405
Stemming	164405
Tokenizing	164405
Stopwords Removal	103413

Pada tahapan *data cleaning* terjadi penambahan kata sebanyak 482 karena pada tahapan ini karakter, angka, simbol dan *hyperlink* dihilangkan sehinga terjadi pengurangan maupun peambahan jumlah kata. Kemungkinan terjadi penambahan kata karena terdapat kata yang sebelumnya ada simbol atau yang lainnya sehingga dihilangkan dan diganti dengan spasi. Oleh karena itu kata sebelumnya bisa terbagi menjadi dua atau lebih kata. Untuk tahapan *case folding* tidak terjadi pengurangan kata pada data karena proses ini hanya menjadikan kata dalam bentuk *lowercase*. Sama seperti sebelumnya, tahapan *stemming* juga tidak mengalami pengurangan kata dalam data dikarenakan hanya menjadikan kata ke bentuk dasar.

Tahapan *tokenizing* juga sama tidak mengalami pengurangan kata pada data karena itu hanya proses pembuatan token dari data yang sudah ada. Selanjutnya pada tahapan *stopwords removal* terjadi pengurangan kata sebanyak 60992 dikarenakan penghapusan kata-kata yang tidak penting berdasarkan daftar *stopwords* yang sudah ditentukan. Adapun kesimpulan yang dapat dilihat pada Tabel 6.1.3.

Tabel 6.1.3 Hasil Pra-Proses Data

	Sebelum Pra-Proses	**Setelah Pra-Proses**
Jumlah Dokumen/Data	5786	5786
Jumlah Kata	163923	103413
Kata Unik	38504	14130

6.1.3 Pembentukan Model Klasifikasi dan Validasi

Dalam melakukan pembentukan model klasifikasi *random forest*, model melalui tahap validasi dengan cara pengujian percobaan beberapa model menggunakan nilai akurasi, *precision, recall* dan *f-measure*.

Hasil Klasifikasi dengan Parameter Penelitian Sebelumnya

Pada klasifikikasi ini, model yang dibangun menggunakan parameter berdasarkan penelitian sebelumnya yang sudah dijelaskan di sub bab 5.5.2. Dari model klasifikasi ini didapatkan hasil akurasi dan *f-mesaure* sebesar 61,56% dan 59,92%. Nilai akurasi tersebut belum dapat diartikan bahwa hasil tersebut memiliki performa yang baik. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision, recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.4.

Tabel 6.1.4 Hasil Evaluasi Performa Klasifikasi Model

Label	**Precision**	**Recall**	**F-Measure**
Berbagi Informasi	68,45%	46,08%	55,08%
Meminta Informasi	69,28%	66,17%	67,69%

Mengemukakan Pendapat	46,66%	85,06%	60,26%
Apresiasi	80,92%	70,49%	75,34%
Komplain Layanan Pemerintahan	82,00%	27,52%	41,21%
AVG/TOTAL	69,46%	59,06%	59,92%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 69,46% dan *recall* sebesar 59,06%. Dari hasil tersebut dapat diambil kesimpulan bahwa performa atau tingkat efektifitas klasifikasi model bisa dikatakan kurang baik karena nilai yang didapatkan cukup rendah. Beberapa faktor yang menyebabkan hasil tersebut salah satunya adalah jumlah data yang digunakan, serta *feature* yang digunakan masih terlalu umum.

Hasil Klasifikasi Percobaan 1

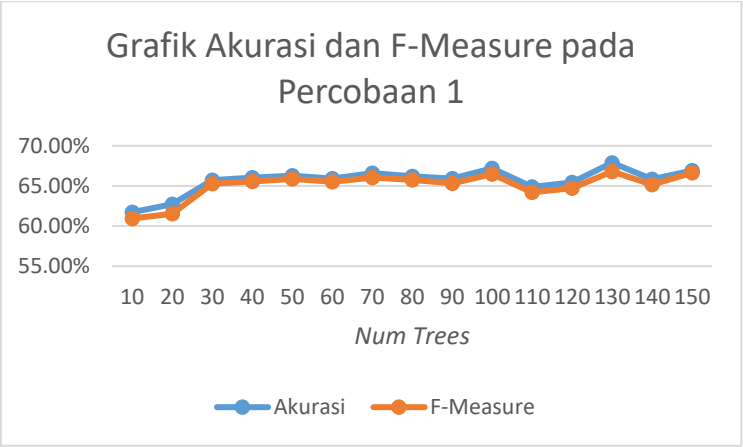
Pada proses klasifikasi ini, percobaan yang dilakukan adalah dengan menggunakan *num of trees* atau jumlah banyak pohon keputusan. Percobaan dilakukan sebanyak 15 kali dengan nilai dasar 10 yang sudah dijelaskan pada sub bab 5.5.2. Interval yang digunakan pada percobaan ini sebesar 10 dan menggunakan 16 *max depth*. Hasil dari setiap iterasi dapat dilihat pada Tabel 6.1.5.

Tabel 6.1.5 Perbandingan Evaluasi Performa Klasifikasi Percobaan 1

Num of Trees	**Uji Validasi**		
	Precision	**Recall**	**F-Measure**
10	67,03%	59,78%	60,95%
20	68,04%	60,62%	61,54%
30	72,51%	63,75%	65,28%
40	71,99%	64,15%	65,58%
50	72,15%	64,39%	65,88%
60	73,13%	63,98%	65,53%
70	73,31%	64,58%	66,02%
80	73,26%	64,17%	65,77%
90	72,45%	63,91%	65,34%
100	72,94%	65,22%	66,49%

110	71,05%	62,86%	64,22%
120	71,66%	63,43%	64,74%
130	72,47%	66,06%	66,80%
140	72,21%	63,84%	65,17%
150	73,39%	65,11%	66,63%

Grafik nilai akurasi dan *f-measure* dari setiap percobaan dapat dilihat pada Gambar 6.1.2.



Gambar 6.1.2 Perbandingan Akurasi dan F-Measure Percobaan 1

Hasil akurasi dan *f-measure* terbaik didapatkan yaitu 67,89% dan 66,80% dengan menggunakan parameter banyak pohon keputusan sebesar 130 pada maksimal kedalaman sebesar 16. Percobaan ini dilakukan sebanyak 2 kali untuk mengetahui apakah pada pohon keputusan 100 dan 130 hanya sebuah *spike* atau tidak. Dari kedua percobaan memberikan hasil yang hampir sama dan menandakan bahwa angka tersebut bukan hanya sebuah *spike*. Nilai tersebut terus mengalami kenaikan dan penurunan baik pada akurasi maupun *f-measure*. Parameter *num of trees* sebesar 130 akan dijadikan dasar pada percobaan selanjutnya. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision*, *recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.6.

Tabel 6.1.6 Hasil Evaluasi Performa Klasifikasi Model Percobaan 1

Label	Precision	Recall	F-Measure
Berbagi Informasi	77,33%	56,86%	65,54%
Meminta Informasi	71,58%	81,44%	76,19%
Mengemukakan Pendapat	55,09%	84,60%	66,73%
Apresiasi	78,88%	68,48%	73,31%
Komplain Layanan Pemerintahan	79,45%	38,93%	52,25%
AVG/TOTAL	72,47%	66,06%	66,80%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 72,47% dan *recall* sebesar 66,06%. Dari hasil tersebut dapat diambil kesimpulan bahwa performa atau tingkat efektifitas klasifikasi model bisa dikatakan belum cukup baik karena rasio nilai *precision* dan *recall* belum seimbang dan nilai di beberapa kategori label cukup rendah walaupun rata-rata atau total nilai sudah cukup baik.

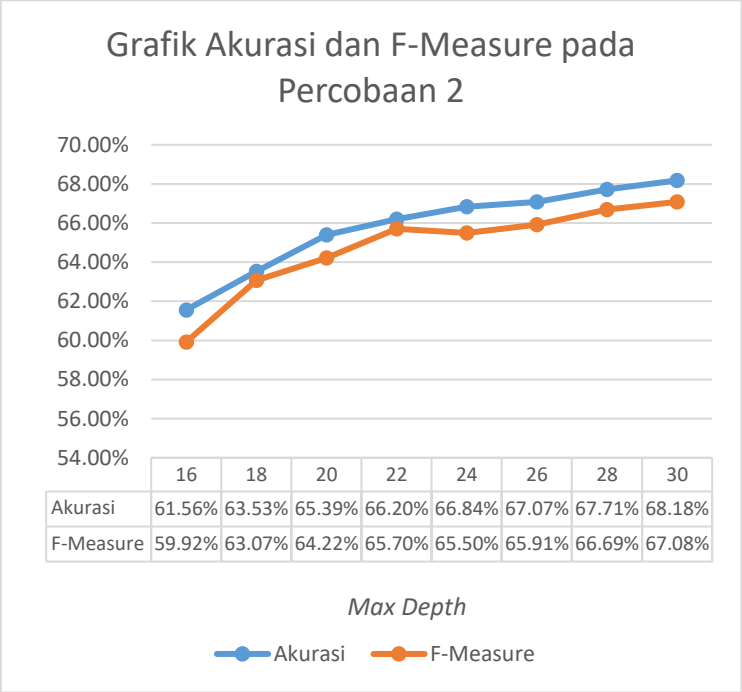
Hasil Klasifikasi Percobaan 2

Pada proses klasifikasi ini, percobaan yang dilakukan adalah dengan menggunakan *max depth* atau maksimal kedalaman. Percobaan dilakukan sebanyak 8 kali dengan interval sebesar 2 dan menggunakan 9 *num of trees*. Hasil dari setiap iterasi dapat dilihat pada Tabel 6.1.7.

Tabel 6.1.7 Perbandingan Evaluasi Perrforma Klasifikasi Percobaan 2

Max Depth	Uji Validasi		
	Precision	**Recall**	**F-Measure**
16	69,46%	59,06%	59,92%
18	68,82%	61,86%	63,07%
20	69,66%	63,89%	64,22%
22	70,46%	64,72%	65,70%
24	69,60%	65,45%	65,50%
26	69,43%	65,67%	65,91%
28	69,39%	66,49%	66,69%
30	70,25%	67,02%	67,08%

Grafik nilai akurasi dan *f-measure* dari setiap percobaan dapat dilihat pada Gambar 6.1.3.



Gambar 6.1.3 Perbandingan Akurasi dan F-Measure Percobaan 2

Hasil akurasi dan *f-measure* terbaik didapatkan yaitu 68,18% dan 67,08% dengan menggunakan parameter maksimal kedalaman sebesar 30 pada banyak pohon keputusan sebesar 9. Nilai tersebut mengalami kenaikan dari parameter awal hingga akhir iterasi. Hal ini menunjukkan bahwa semakin besar *max depth* yang diinisai pada setiap iterasi, semakin baik akurasi yang didapatkan. Parameter *max depth* sebesar 30 akan dijadikan dasar pada percobaan selanjutnya. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision*, *recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.8.

Tabel 6.1.8 Hasil Evaluasi Performa Klasifikasi Model Percobaan 2

Label	Precision	Recall	F-Measure
Berbagi Informasi	74,14%	63,73%	68,54%
Meminta Informasi	65,79%	82,34%	73,14%
Mengemukakan Pendapat	60,89%	75,17%	67,28%
Apresiasi	74,14%	73,93%	74,03%
Komplain Layanan Pemerintahan	76,28%	39,93%	52,42%
AVG/TOTAL	70,25%	67,02%	67,08%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 70,25% dan *recall* sebesar 67,02%. Dari hasil tersebut dapat diambil kesimpulan bahwa performa atau tingkat efektifitas klasifikasi model bisa dikatakan masih kurang baik karena nilai yang didapatkan lebih rendah dibandingkan percobaan ke-1.

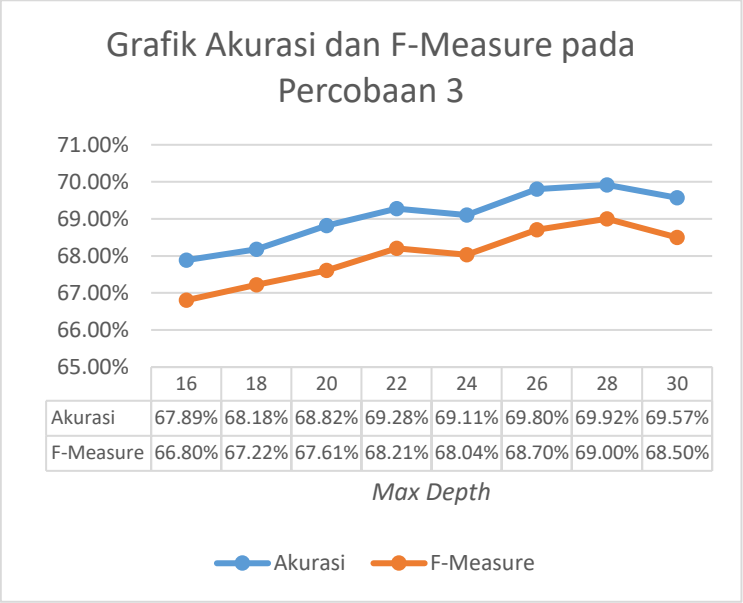
Hasil Klasifikasi Percobaan 3

Percobaan proses klasifikasi ini menggunakan *max depth* atau maksimal kedalaman yang dilakukan sebanyak 8 kali dengan interval sebesar 2 dan menggunakan 130 *num of trees* yang merupakan hasil terbaik dari percobaan ke-1. Hasil dari setiap iterasi dapat dilihat pada Tabel 6.1.9.

Tabel 6.1.9 Perbandingan Evaluasi Perrforma Klasifikasi Percobaan 3

Max Depth	Uji Validasi		
	Precision	**Recall**	**F-Measure**
16	72,47%	59,06%	66,80%
18	72,69%	61,86%	67,22%
20	72,41%	63,89%	67,61%
22	72,37%	64,72%	68,21%
24	72,16%	65,45%	68,04%
26	72,90%	65,67%	68,70%
28	72,59%	66,49%	69,00%
30	72,03%	67,02%	68,50%

Grafik nilai akurasi dan *f-measure* dari setiap percobaan dapat dilihat pada Gambar 6.1.4.



Gambar 6.1.4 Perbandingan Akurasi dan F-Measure Percobaan 3

Hasil akurasi dan *f-measure* terbaik didapatkan yaitu 69,92% dan 68,99% dengan menggunakan parameter maksimal kedalaman sebesar 28 pada banyak pohon keputusan sebesar 130. Nilai tersebut mengalami kenaikan sampai iterasi ke-4 yang selanjutnya mengalami penurunan pada iterasi ke-5. Selanjutnya nilai tersebut mengalami kenaikan lagi sampai iterasi ke-7 yang merupakan hasil terbaik pada percobaan ini lalu mengalami penurunan pada iterasi terakhir. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision*, *recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.10.

Tabel 6.1.10 Hasil Evaluasi Performa Klasifikasi Model Percobaan 3

Label	Precision	Recall	F-Measure
Berbagi Informasi	75,63%	68,95%	72,14%

Meminta Informasi	69,49%	81,14%	74,86%
Mengemukakan Pendapat	60,00%	78,62%	68,06%
Apresiasi	78,25%	74,21%	76,18%
Komplain Layanan Pemerintahan	79,61%	40,60%	53,78%
AVG/TOTAL	72,59%	66,49%	69,00%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 72,59% dan *recall* sebesar 66,49%. Dari hasil tersebut dapat diambil kesimpulan bahwa performa atau tingkat efektifitas klasifikasi model bisa dikatakan belum cukup baik karena nilai yang didapatkan belum cukup seimbang.

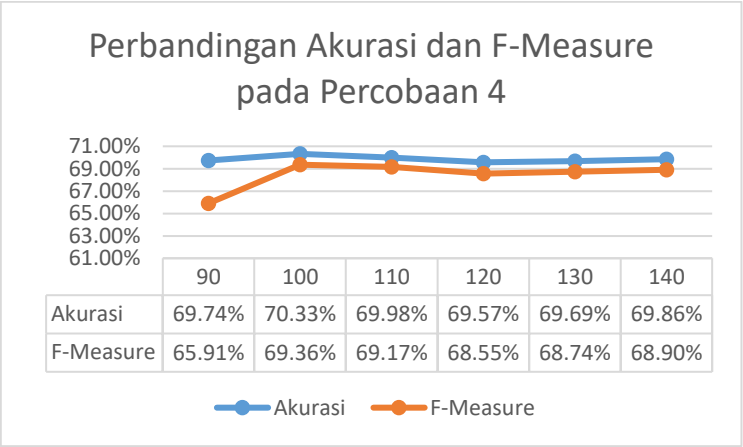
Hasil Klasifikasi Percobaan 4

Percobaan proses klasifikasi ini menggunakan *num of trees* atau jumlah banyak pohon keputusan yang dilakukan sebanyak 6 kali yaitu dari 90 sampai 140 yang diambil dari percobaan pertama yang menunjukkan hasil dari interval dua nilai tertinggi ditambah dua nilai sebelum tertinggi petama dan sesudah tertinggi kedua. Interval yang digunakan masih sama yaitu 10 dan menggunakan 30 *max depth*. Hasil dari setiap iterasi dapat dilihat pada Tabel 6.1.11.

Tabel 6.1.11 Perbandingan Evaluasi Perrforma Klasifikasi Percobaan 4

Num of Trees	**Uji Validasi**		
	Precision	**Recall**	**F-Measure**
90	71,71%	68,53%	65,91%
100	73,14%	69,02%	69,36%
110	72,34%	68,84%	69,17%
120	72,05%	68,27%	68,55%
130	72,23%	68,43%	68,74%
140	72,44%	69,72%	68,90%

Grafik nilai akurasi dan *f-measure* dari setiap percobaan dapat dilihat pada Gambar 6.1.5.



Gambar 6.1.5 Perbandingan Akurasi dan F-Measure Percobaan 4

Hasil akurasi dan *f-measure* terbaik didapatkan yaitu 70,33% dan 69,46% dengan menggunakan parameter banyak pohon keputusan sebesar 100 pada maksimal kedalaman sebesar 30. Nilai tersebut mengalami penurunan dari nilai awal hingga akhir iterasi. Hasil ini merupakan hasil terbaik yang didapatkan dari 4 percobaan yang dilakukan. Parameter *num of trees* sebesar 100 dan *max depth* sebesar 30 akan dijadikan dasar pada percobaan selanjutnya dengan data yang lebih terdistribusi secara merata. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision*, *recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.12.

Tabel 6.1.12 Hasil Evaluasi Performa Klasifikasi Model Percobaan 4

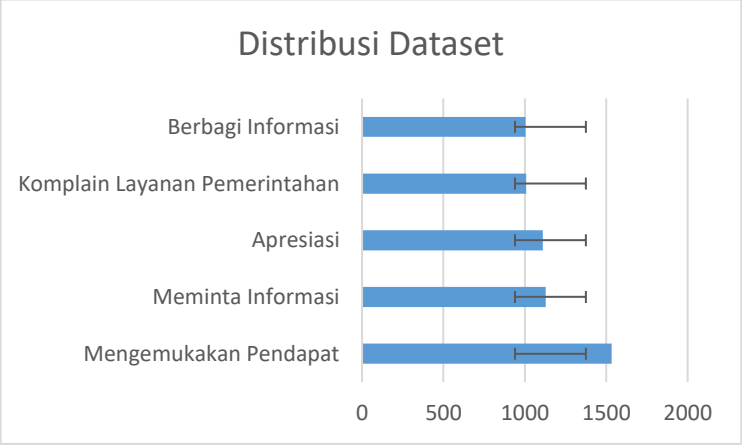
Label	Precision	Recall	F-Measure
Berbagi Informasi	78,03%	67,32%	72,28%
Meminta Informasi	69,17%	82,63%	75,31%
Mengemukakan Pendapat	60,52%	80,00%	68,91%
Apresiasi	78,25%	74,21%	76,18%
Komplain Layanan Pemerintahan	79,74%	40,94%	54,10%

AVG/TOTAL	73,14%	69,02%	69,36%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 73,14% dan *recall* sebesar 69,02%. Dari hasil tersebut dapat diambil kesimpulan bahwa performa atau tingkat efektifitas klasifikasi model bisa dikatakan cukup baik. Nilai tersebut menunjukkan bahwa relevansi atau kecocokan apa yang dicari dengan apa yang ditemukan lebih tinggi dikarenakan jumlah yang relevan lebih besar atau seimbang jika dibandingkan dengan yang tidak relevan walaupun ada beberapa kategori yang masih memiliki nilai yang kurang baik.

Hasil Klasifikasi dengan Proporsi Data Seimbang

Pada percobaan ini, data yang digunakan pada percobaan sebelumnya lebih terdistribusi secara merata. Data sebelumnya yang terdapat satu label diatas standar deviasi yang dapat dilihat pada Gambar 6.1.6. Untuk mengatasi data yang *imbalanced* dilakukan metode undersampling terhadap satu label tersebut agar masih dalam lingkup standar deviasi [34]. Hasil distribusi data yang digunakan pada percobaan ini dapat dilihat pada Tabel 6.1.13.



Gambar 6.1.6 Distribusi Dataset dengan Standar Deviasi

Tabel 6.1.13 Data Pendistribusian Merata

Label	Jumlah Data
Berbagi Informasi	1005
Meminta Informasi	1129
Mengemukakan Pendapat	1186
Apresiasi	1111
Komplain Layanan Pemerintahan	1008
Total	5439

Pembangunan model berdasarkan paramater pada percobaan sebelumnya yang mendapatkan nilai akurasi terbaik yaitu 100 *num trees* dan 30 *max depth.* Hasil akurasi dan *f-measure* yang didapatkan pada percobaan ini lebih baik dibandingkan percobaan sebelumnya yaitu 74,25% dan 73,37%. Setelah itu untuk melihat lebih jauh lagi performa hasil klasifikasi dari model ini, dilakukan penghitungan nilai *precision*, *recall*, dan *f-measure* yang hasilnya dapat dilihat pada Tabel 6.1.14.

Tabel 6.1.14 Hasil Evaluasi Performa Klasifikasi Model Proporsi Data Seimbang

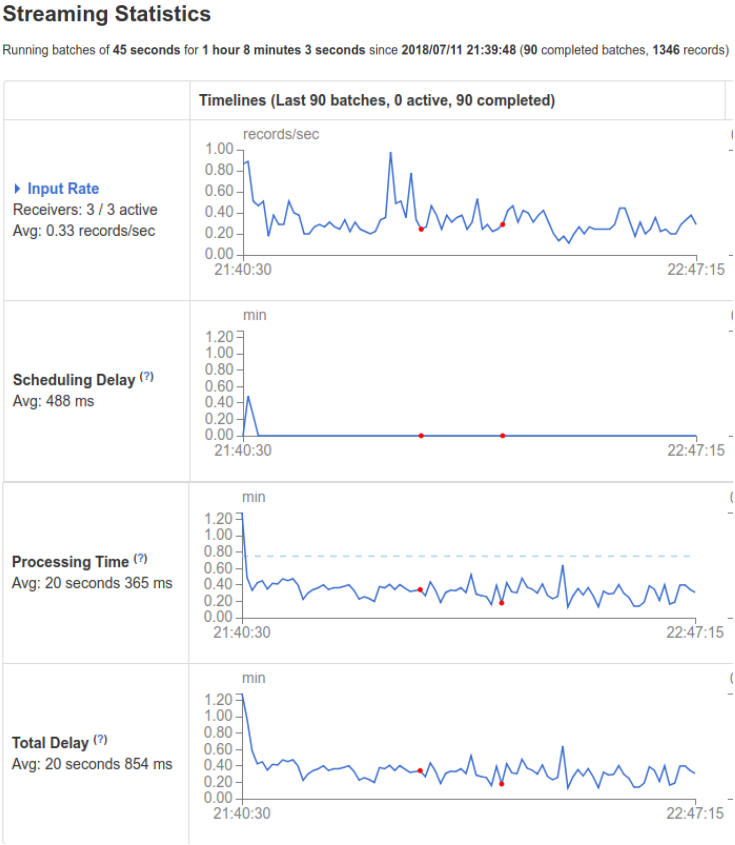
Label	Precision	Recall	F-Measure
Berbagi Informasi	82,11%	65,37%	72,79%
Meminta Informasi	72,98%	85,50%	78,75%
Mengemukakan Pendapat	64,27%	87,86%	74,24%
Apresiasi	83,02%	79,28%	81,11%
Komplain Layanan Pemerintahan	76,84%	49,16%	59,96%
AVG/TOTAL	75,85%	73,43%	73,37%

Dari penghitungan performa tersebut didapatkan hasil *precision* sebesar 75,85% dan *recall* sebesar 73,43%. Dari hasil tersebut dapat diambil kesimpulan bahwa penggunaan data yang lebih merata pada pembuatan model klasifikasi menggunakan metode *random forest* dapat meningkatkan performas model. Hasil tersebut juga menunjukkan performa atau tingkat efektifitas klasifikasi model bisa dikatakan sudah baik karena

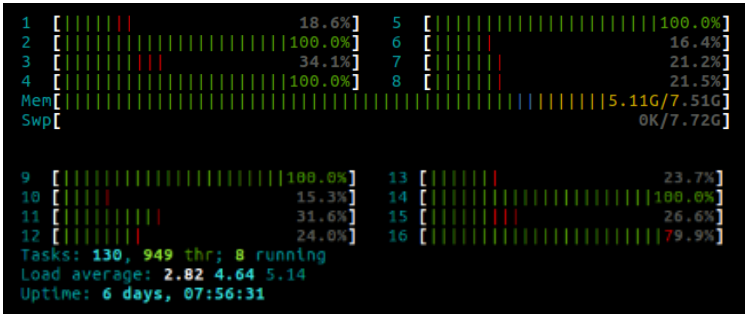
nilai anatara *precision* dan *recall* sudah cukup seimbang. Nilai tersebut juga menunjukkan nilai yang jauh lebih baik dari percobaan sebelumnya. Namun ada satu kategori label yang memiliki nilai *recall* jauh dibawah yang diharapkan, hal ini bisa terjadi karena dataset yang digunakan pada saat proses pemberian label sangat bias sehingga kurang mempresentasikan label tersebut. Model ini dijadikan model klasifikasi yang digunakan pada aplikasi untuk mengklasifikasikan data komentar netizen melalui proses *streaming.*

6.2 Pengujian Aplikasi

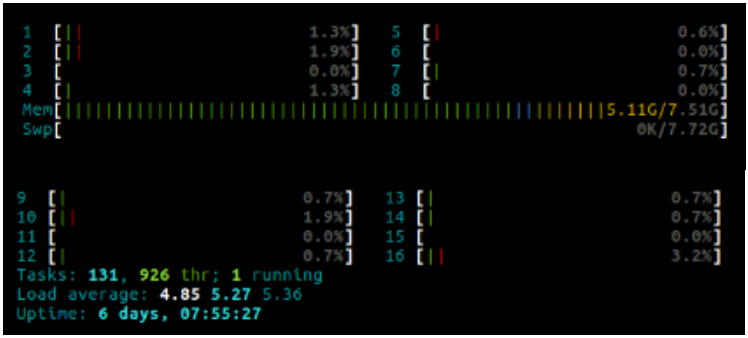
Pengujian aplikasi ini dengan cara melakukan tes integrasi pada aplikasi dengan melihat kerja dari model yang telah dibangun untuk melakukan klasifikasi pada situs Egovbench. Pada percobaan penerapan integrasi dilakukan menjalankan aplikasi *streaming* data kemudian melakukan pengecekan pada Spark *web ui* di *host* server untuk melihat statistik streaming dari *Input Rate* dari ketiga *receiver, Scheduling Delay*, *Processing Time* dan *Total Delay*. Hasil rata-rata dari *Input Rate* adalah 0,33 record/detik. Sedangkan hasil rata-rata *Scheduling Delay* adalah 488 milidetik. Untuk rata-rata *Processing Time* adalah 20,385 detik dan rata-rata *Total Delay* adalah 20,854 detik yang dapat dilihat pada Gambar 6.2.1. Disamping itu juga melakukan pengecekan pada kinerja CPU yang pada Gambar 6.2.3 menunjukkan pada saat proses data dan Gambar 6.2.2 menunjukkan pada saat *batching* data. Proses *streaming* ini memakan memori rata-rata sebesar 5gb. Dalam pengujian aplikasi *streaming* ini mennjukkan bahwa sistem *micro-batching* sangat banyak keunggulan dalam hal menangani data input dari Kafka sehingga tidak *overload* dan tidak terlalu memakan memori yang cukup besar. Selain itu kerja CPU tidak dipaksa untuk bekerja secara terus menerus pada data yang cukup cepat. Interval *batch* sudah optimal tanpa adanya delay proses batch selanjutnya.



Gambar 6.2.1 Statistik Streaming



Gambar 6.2.3 CPU pada proses data



Gambar 6.2.2 CPU pada *batching* data

Halaman ini sengaja dikosongkan

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan dan saran dalam pengerjaan tugas akhir. Kesimpulan ini diharapkan dapat menjawab tujuan dari penelitian. Saran dapat digunakan oleh peneliti selanjutnya.

7.1 Kesimpulan

Berdasarkan proses-proses yang telah dilakukan dalam pengerjaan tugas akhir dengan judul "Rancang Bangun Aplikasi untuk Klasifikasi Komentar Netizen pada Media Sosial Pemerintah Daerah di Indonesia Menggunakan Algoritma *Random Forest*" yang telah dilakukan, dapat disimpulkan sebagai berikut:

1. Penelitian ini membuktikan bahwa metode *crawling* dan *streaming* data dengan memanfaatkan *API* dari masing-masing media sosial mampu menjalankan proses pengumpulan data.
2. Hasil dari pengumpulan data dengan cara *crawling* data komentar dari 867 akun pemda diantaranya 149 akun resmi dan 164 akun influencer dari media sosial Facebook, 202 akun resmi dan 179 akun influencer dari media sosial Twitter, 113 akun resmi dan 60 akun influencer dari media sosial Youtube, didapatkan data sebesar 1400396 dengan rincian 1084013 dari Facebook, 136718 dari Twitter, dan 179665 dari Youtube.
3. Pemodelan Random Forest dapat digunakan untuk melakukan klasifikasi komentar netizen dengan label tertentu sesuai dengan hasil penelitian sebelumnya. Pada percobaan ke-2 membuktikan bahwa paramter *max depth* lebih berpengaruh daripada *num of trees* terhadap nilai akurasi yang didapatkan.

4. Hasil terbaik adalah menggunakan data sebanyak 5490 dengan menggunakan parameter 100 pohon keputusan dan 30 maksimal kedalaman. Hasil tersebut mendapatkan nilai akurasi dan *f-measure* sebesar 74,25% dan 73,37% dengan rata-rata *precision* dan *recall* sebesar 73,43% dan 73,37%. Hal ini membuktikan bahwa dengan distribusi label yang secara merata pada metode klasifikasi *random forest* lebih mendapatkan hasil yang lebih baik.
5. Perancangan aplikasi Egovbench dapat mengintegrasikan bahasa pemrograman PHP untuk kebutuhan *streaming* dan visualisasi, dan bahasa pemrograman Python untuk pra-proses data, hingga klasifikasi data mampu menjawab kebutuhan visualisasi komentar netizen pada pemerintah daerah.

7.2 Saran

Dalam pengerjaan tugas akhir ini masih terdapat banyak kekurangan yang dapat diperbaiki. Saran ini diharapkan dapat membantu agar hasil penelitian kedepannya dapat lebih baik. Saran penulis untuk penelitian selanjutnya sebagai berikut:

1. Percobaan pembuatan model klasifikasi dengan pembagian proporsi data latih dan data uji yang berbeda.
2. Pelabelan dataset dilakukan lebih dari 1 annotator untuk mengurangi data yang bias.
3. Meningkatkan dan menggunakan data yang lebih terdistribusi secara merata dari sisi jumlah label kategori.
4. Melakukan percobaan dengan menggunakan metode klasifikasi yang berbeda untuk melihat kemungkinan akurasi yang lebih baik.
5. Untuk aplikasi Egovbench kedepannya perlu dilakukan penelitian lebih lanjut mengenai *dynamic classification*

model, sehingga pembentukan model akan berjalan otomatis pada setiap data yang sudah masuk melalui proses *streaming* dan memperkaya fitur aplikasi dalam analisa sosial media.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Presiden Republik Indonesia, "Instruksi Presiden Republik Indonesia Nomor 3 Tahun 2003 Tentang Kebijakan dan Strategi Nasional Pengembangan E-Government," 2003.

[2] S. Kemp, "Digital in 2018," 2018. [Online]. Available: https://hootsuite.com/pages/digital-in-2018. [Accessed: 08-Feb-2018].

[3] D. Purworini, "Model Informasi Publik Di Era Media Sosial : Kajian Grounded Teori Di Pemda Sukoharjo," *Model Inf. Publik di Era Media Sos.*, vol. 6, no. 1, pp. 1–12, 2014.

[4] A. N. Abadi, *Rancang Bangun Perangkat Lunak Benchmarking Sosial Media Pemerintah Daerah Indonesia*. 2017.

[5] M. Magnusson, P. Bellström, and C. Thoren, "Facebook usage in government – a case study of information content," *AMCIS 2012 Proc.*, pp. 1–10, 2012.

[6] M. Al-janabi and P. Andras, "A Systematic Analysis of Random Forest Based Social Media Spam Classification," *Netw. Syst. Secur.*, vol. 10394, pp. 427–438, 2017.

[7] N. Wang, B. Varghese, and P. D. Donnelly, "A Machine Learning Analysis of Twitter Sentiment to the Sandy Hook Shootings," 2016.

[8] T. Almarabeh and A. Abuali, "A General Framework for E-Government: Definition Maturity Challenges, Opportunities, and Success," *Eur. J. Sci. Res.*, vol. 39, no. 1, pp. 1450–216, 2010.

[9] V. Ndou, "E – Government for Developing Countries: Opportunities and Challenges," *EJISDC*

Electron. J. Inf. Syst. Dev. Ctries., vol. 18, no. 1, pp. 1–24, 2004.

[10] M. Dewing, "Social Media : An Introduction Social Media : An Introduction," *Libr. Parliam.*, no. 2010, pp. 1–2, 2012.

[11] V. Taprial and P. Kanwar, *Understanding Social Media*, no. 1. 2014.

[12] "EGov Benchmark." [Online]. Available: http://egovbench.addi.is.its.ac.id/methodology.php. [Accessed: 28-Jan-2018].

[13] Kementrian Pendayagunaan Aparatur Negara dan Reformasi Birokrasi Republik Indonesia, "Pedoman Pemanfaatan Media Sosial Instansi Pemerintah," 2012.

[14] M. Najork, "Web crawler architecture," *Encycl. Database Syst.*, pp. 3–5, 2009.

[15] C. Olston and M. Najork, "Web Crawling," *Found. Trends® Inf. Retr.*, vol. 4, no. 3, pp. 175–246, 2010.

[16] C. Castillo, "Effective web crawling," *ACM SIGIR Forum*, vol. 39, no. 1, p. 55, 2005.

[17] N. Garg, *Apache Kafka*. 2013.

[18] Apache Kafka, "Introduction," 2017. [Online]. Available: https://kafka.apache.org/intro.html. [Accessed: 21-Feb-2018].

[19] M. Gupta and N. Aggarwal, "Classification Techniques Analysis," *Natl. Conf. Comput. Instrum.*, pp. 128–131, 2010.

[20] S. Neelamegam and E. Ramaraj, "Classification algorithm in Data mining : An Overview," vol. 4, no. 8, pp. 369–374, 2013.

[21] W. T. Aung and K. H. M. S. Hla, "Random forest classifier for multi-category classification of web pages," *2009 IEEE Asia-Pacific Serv. Comput. Conf. APSCC 2009*, pp. 372–376, 2009.

[22] M. Pal, “Random forest classifier for remote sensing classification,” *Int. J. Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2005.

[23] A. Liaw and M. Wiener, “Classification and Regression by randomForest,” *R News*, vol. 55, no. 2/3, pp. 18–22, 2002.

[24] S. R. Joelsson, J. A. Benediktsson, and J. R. Sveinsson, “Random forest classifiers for hyperspectral data,” *Int. Geosci. Remote Sens. Symp.*, vol. 1, pp. 160–163, 2005.

[25] D. M. W. POWERS, “Evaluation: From Precision, Recall and F-Measure To Roc, Informedness, Markedness & Correlation,” *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[26] C. Goutte and E. Gaussier, “A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation,” vol. 3408, pp. 345–359, 2005.

[27] M. Klassen and N. Paturi, “Web document classification by keywords using random forests,” *Networked Digit. Technol.*, pp. 256–261, 2010.

[28] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, “ReVision: Automated Classification, Analysis and Redesign of Chart Images,” *Proc. 24th Annu. ACM Symp. User interface Softw. Technol. (UIST ’11)*, pp. 393–402, 2011.

[29] J. Geovedi, “Indonesian wordlist.” [Online]. Available: https://github.com/geovedi/indonesian-wordlist. [Accessed: 04-Apr-2018].

[30] F. Z. Tala, “A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia,” *M.Sc. Thesis, Append. D*, vol. pp, pp. 39–46, 2003.

[31] G. K. Prakoso, *Rancang Bangun Apliaksi untuk*

Klasifikasi Post pada Sosial Media Pemerintah Daerah di Indonesia Menggunakan Support Vector Machine (SVM). 2018.

[32] D. Hanindito, *Rancang bangun Sistem Penilaian Reaksi Masyarakat Terhadap Akun Sosial Media Pemerintah Daerah*. 2018.

[33] N. R. Pratomo, *Rancang Bangun Aplikasi untuk Kategorisasi Komentar Netizen pada Media Sosial Pemerintah Daerah Terhadap SKPD*. 2018.

[34] D. Ramyachitra and P. Manikandan, "IMBALANCED DATASET CLASSIFICATION AND SOLUTIONS : A REVIEW," vol. 5, no. 4, 2014.

BIODATA PENULIS

Penulis lahir di kota Bangil pada tanggal 10 April 1996. Anak ketiga dari tiga bersaudara yang telah menempuh pendidikan formal yaitu; SD Negeri 1 Plus Bangil, SMP Negeri 2 Bangil, dan SMA Negeri 1 Bangil.

Pada tahun 2014, penulis melanjutkan pendidikan ke jenjang yang lebih tinggi, yaitu di Institut Teknologi Sepuluh Nopember (ITS) Surabaya, sebagai mahasiswa departemen Sistem Informasi, Fakultas Tekonologi Informasi dan Komunikasi (FTIK). Terdaftar sebagai pemilik NRP 0511440000143. Selama menjadi mahasiswa, penulis banyak mengikuti kegiatan kemahasiswaan, antara lain seminar, organisasi dan kajian. Penulis juga aktif dalam keorganisasian sebagai staff *External Affairs* (EA) BEM FTIf periode 2015/2016 dan sebagai staff ahli *Affairs* (EA) BEM FTIf periode 2016/2017. Disamping aktif dalam kegiatan kemahasiswaan, penulis juga pernah menjadi Asisten Praktikum mata kuliah Desain Manajemen Jaringan Komputer. Pada tahun ke-4, penulis tertarik dengan bidang Social Media Analysis, sehingga mengambil bidang minat laboratorium Akuisisi Data dan Diseminasi Informasi (ADDI) dan lulus dalam waktu 4 tahun atau 8 semester. Penulis dapat dihubungimelalui email m.fikryhazmi@gmail.com